Massachusetts Inst. of Technology, Cambridge

Establishment and operation of a New England Center at MIT for Computer Technology and Research in Biomedical Sciences

TSI Tin initial caps: FINAL REPORT

(Contract PH43-63-540)

National Institutes of Health, Bethesda, Md.

(2) OTS

W. A. Clark (MIT) et al

October 29, 1963 158 p ref

~~Cambridge, Massachusetts~~

(NASA Order R-78; Contract PH43-63-540)

(NASA CR-53599)

OTS: $11.50 pd

PART I

INTRODUCTION

Research in the life sciences typically deals with systems that are non-linear, multicomponent, multipurpose and hence multivariate. Non-trivial interactions between components are the rule. In addition, studies of biological systems frequently yield large volumes of data, generated at high rates. These factors, singly and in combination, constitute a formidable challenge to mathematical analysis or synthesis; they largely account for the fact that much of systems biology has had to content itself with descriptive, phenomenological statements.

High-speed digital computers and the associated information-processing technology embody means for realistically coming to grips with some of the obstacles to a more quantitative, theory-oriented biology. In order for these computer complexes to be maximally effective, cognizance must be taken of the special character of research in the biomedical sciences. Such research poses requirements of size, speed and versatility that can only be met by computers whose design puts a premium on machine capability and operational flexibility.

One need in biomedical computing, hitherto largely unsatisfied, is basically for machines and systems capable of processing and manipulating large volumes of complex data, at speeds compatible with on-going experimentation. The stipulation of "real-time," "on-line" operation derives from the frequent necessity for immediate feedback between experimenter and data. Computers are thus viewed as tools in experimentation, not unlike their use in process control. Computers as laboratory instruments need to be responsive to frequent changes in experimental procedures and to the peculiar input-output requirements that characterize much of biomedical research.

In addition to data processing and analysis, modeling and simulation of biological systems can be carried out with the aid of computer systems having the requisite flexibility and power.

Most currently available computer equipment fails to fulfill several of the diverse requirements of the biomedical laboratory. Successful design and development of computer systems geared to the needs of the life scientist can only evolve from active, day-by-day collaboration of computer technologists with biomedical investigators. Hence the need for a Center joining computer technology to research in the biomedical sciences.

CHRONOLOGY

Relevant Events Leading to the Concept of a New England Center for Computer Technology and Research in the Biomedical Sciences

1953 Time-Gated Amplitude Quantizer for neural signals (K. Putter). First specialized computing device in CBG-RLE for bioelectric activity.

Farley memo on potential use of computers in neurophysiology and biomedical work.

Clark and Farley initiated studies of neuron-like networks at Lincoln Laboratory.

1954 Analog Correlator System for brain potentials (J. S. Barlow and R. M. Brown). Constructed at RLE in collaboration with Dr. M. A. B. Brazier's group at Massachusetts General Hospital.

1955 Evoked Response Detector for brain potentials constructed at RLE. (J. S. Barlow and G. Fahringer).

1956 Special-purpose computing device, Amplitude and Latency Measuring Instrument with Digital Output (ALMIDO), constructed at RLE (R. L. Koehler).

1957 Design of Average Response Computer (ARC-1) by W. A. Clark at Lincoln Laboratory.

Studies of patterns in the electroencephalogram with the aid of the Lincoln Laboratory TX-0 computer* (jointly by members of DCG and CBG).

Proposal by Papian and Clark (DCG) to put TX-0 on MIT campus for use by CBG.

1958 TX-0 transferred from Lincoln Laboratory to MIT's Department of Electrical Engineering for use in teaching and research.

TX-2 designed and constructed at Lincoln Laboratory under the direction of W. A. Clark. Design influenced in part by expected use for biological data processing and simulation of neuron-like networks.

* TX-0 computer designed and constructed in 1955 by Papian, Clark, et al.

1959 Two-week special Summer Session Program at MIT on Quantitative Approaches to the Study of Neuroelectric Activity. Under the direction of W. A. Rosenblith and M. H. Goldstein, Jr. (Department of Electrical Engineering and CBG). Lectures and demonstrations presented by members of CEG and DCG.

1960 Discussion of DCG-CBG collaboration in the design of a programmed stimulus generator resulted in Clark proposal for design and construction of a laboratory-type, general-purpose digital computer.

1961 First demonstration of prototype Laboratory Instrument Computer (LINC) designed at Lincoln Laboratory by Clark and Lt. C. E. Molnar, Air Force Cambridge Research Laboratory (formerly with DCG and CBG).

NSF-sponsored Research Training Conference in Computer Techniques for Biological Scientists (3 weeks) organized by committee composed of W. A. Clark, J. B. Dennis (Department of Electrical Engineering), M. Eden, W. M. Siebert (Department of Electrical Engineering) and T. T. Sandel, Chairman. Computer chiefly used: TX-0.

1962 2nd NSF-sponsored Research Training Conference in Computer Techniques for Biological Scientists organized by J. B. Dennis, M. Eden and T. T. Sandel, Chairman. Computer chiefly used: LINC.

Events Preceding Formation of Center

During 1961 the management of MIT's Lincoln Laboratory indicated that in view of its mission-orientation, major commitments of facilities and personnel to health science problems were not appropriate. Members of the Digital Computer Group (DCG) at Lincoln who had been cooperating with the Communications Biophysics Group (CBG) at MIT's Research Laboratory of Electronics then initiated efforts to determine whether they could find support for a large-scale effort within the health science community.

Informal conversations were held with members of the faculty and administration of MIT and other academic institutions, as well as with administrators at NIH. In 1959 the National Institutes of Health had formed an Advisory Committee on Computers in Research (ACCR) to act as a study section and to explore the usefulness of computers in biology and medicine. As part of ACCR's January 1962 report to the Director of NIH, W. N. Papian (then group leader of the DCG) prepared an appendix entitled: Model for a Biomedical Computing Center which is oriented toward Computer Technology. With the aid of this document and one prepared by M. Eden of MIT's Electrical Engineering Department, W. A. Clark (DCG), M. Eden, B. G. Farley (DCG), W. N. Papian and T. T. Sandel (DCG) drafted a "working paper" in July 1962 for a Center for Computer Technology in the Life Sciences.

This working paper was submitted for appraisal to Dr. C. H. Townes (Provost at MIT) who then appointed a committee to report on the possible establishment of such a Center at MIT. Members of the committee were W. A. Clark, M. Eden, P. Elias (Head, Department of Electrical Engineering), W. N. Papian, F. O. Schmitt (MIT Institute Professor), and W. A. Rosenblith (Department of

Appendix 9

Electrical Engineering and Group Leader of CBG). The committee submitted a favorable report to Dr. Townes in October 1962.

The MIT administration designated W. A. Rosenblith to be responsible for the preparation of a proposal to NIH for financial support of a LINC development program and for negotiations with NIH regarding the establishment of a Center for Computer Technology and Research in the Biomedical Sciences.

In February 1963 Dr. Townes invited representatives of a group of New England institutions to meet to discuss formation of a regional center involving both computer technology and the biomedical sciences. Subsequent to a meeting on February 28th, the interested institutions designated representatives to serve on several committees concerned with the preparation of a proposal to the National Institutes of Health. The following committees were active during March and the first half of April, 1963:

1. Institutional relations
2. Scientific advisory and program
3. Facilities and space requirements
4. Hospital advisory
5. Training programs

On April 15, 1963, the Proposal was presented to the National Institutes of Health.

SCOPE OF OPERATIONS

Conceptually, a Regional Center at the interface of computer technology and the biomedical sciences has two major objectives: (1) it should serve as a resource of facilities and talent for the participating academic and research institutions and (2) it should foster programs in computer design and development coupled to a broad-scale effort of basic research in the life sciences. The success of the enterprise will be largely predicated on the extent to which these twin programs -- the technological and the scientific -- complement and enhance one another. These considerations are reflected in the definition of the primary functions of the Center for Computer Technology and Research in the Biomedical Sciences, namely:

1. To conduct research in computer technology and to develop computer systems appropriate to problems in the biological and medical sciences, and
2. To conduct coherent research programs in those areas of the biological and medical sciences in which the life scientists and the computer scientists involved have overlapping interests.

The implementation of the first function calls for a series of broadly-conceived projects in such diverse aspects of computer technology as systems design, programming and logic studies, memory engineering, development of circuit and fabrication techniques, and materials and components research. Within the range of biological and medical research activities encompassed by the second of the above-listed functions, it is contemplated that initial efforts would emphasize studies of the nervous system. This emphasis is a

natural outgrowth of the past concern and experience of the Center's initial staff with the application of quantitative methods to the study of neuroelectric phenomena. Other areas of research in the life science would presumably benefit from the use of computer techniques, especially those problems that involve real-time processing of large volumes of data, the use of computers as laboratory instruments, or theoretical formulations not readily amenable to mathematical analysis.

The Center would operate a number of special computer facilities, eventually including a new large computer system of the TX-2 variety as well as intermediate-size computers and small laboratory instrument computers such as LINC. It would also develop and maintain special biological facilities and equipment essential to the on-going research programs. These facilities would be designed primarily to serve the needs of the intramural research and development programs, but could be made available for appropriate uses in the participating institutions.

From its role as a regional resource in an academic setting, the following additional functions derive:

3. To provide facilities for faculty affiliates and post-doctoral fellows.
4. To afford opportunities for research by graduate students from participating institutions.
5. To conduct training programs for workers in biomedical and computer research.
6. To carry out advisory functions that fall within the scope of the Center's activities. (The Center's facilities would not be used for services already provided by existing computation centers.)

As soon as practicably feasible, the Center would seek to provide suitable facilities for faculty affiliates from the participating institutions as well as for qualified post-doctoral fellows. The latter, comprising both biomedical investigators and computer scientists, would most likely be recruited by means of a training fellowship program.

Concomitant with its research and development programs, the Center would try to carry out a variety of educational and advisory functions in relevant fields, geared to the needs of participating institutions. Graduate students would be encouraged to participate in research at the Center, under the active supervision of their academic sponsors and of Center staff members. Additional training activities envisaged invlude: short-term traineeships, intensive workshops, seminars and symposia, special lectures and demonstrations, as well as the preparation of reports, training films and documents of educational value.

As part of the Center's advisory services, staff members with appropriate skill and knowledge would make recommendations concerning the disposition of specific problems brought to them by scientists from participating institutions. For relatively simple problems, an analytic solution or literature references might suffice. For problems of sufficient scope and suitability, the inquirer might be invited, at least temporarily, to participate in the Center's activities.

The role which the Center could play in furthering the progress of clinical research would have to be considered in the light of the competences of the Center staff and of interested workers in clinical research. Clinical research workers with the requisite mathematical sophistication to make effective use of the proposed facilities are not currently available and few

appropriate problems have as yet been formulated. However, clinical research is likely to uncover numerous problems requiring high-rate, experiment-oriented data handling. A Clinical Advisory Committee would be established representing the medical schools and hospitals in the community, in order to identify and select appropriate research problems.

ORGANIZATIONAL STRUCTURE AND COMPOSITION

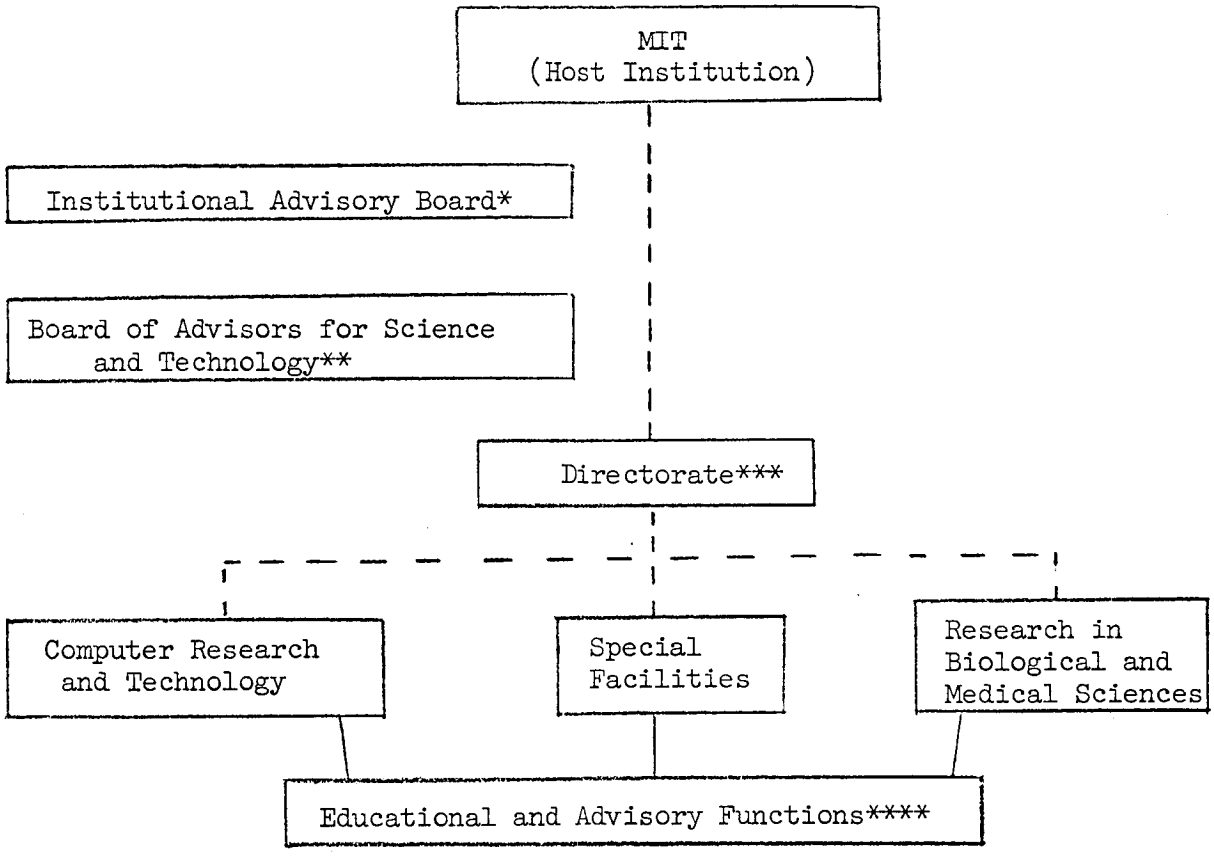
Staffing

The staff of the Center would be composed of computer technologists and scientists, applied mathematicians, as well as physical, biological and medical scientists. It is anticipated that a large proportion of the staff would have some academic affiliation with one of the participating institutions. In this manner the New England institutions could take maximum advantage of the Center as a resource while the Center could maintain effective contact with a broad spectrum of research and education in the biomedical sciences and engineering.

The Center's professional staff would divide into the following categories: (a) scientists and engineers whose institutional attachment would be to the Center only; (b) scientists and engineers whose major commitment would be to the Center but who hold appointments such as Lecturers, Research Associates or Adjunct Professors at the participating institutions; (c) faculty members at the participating institutions with major academic commitments who consider the Center as their, and their students', research home; (d) faculty and staff members of the participating institutions whose primary educational and research commitments are outside the Center but who would expect to participate in the Center's activities, on a part-time or part-year basis (this category would be designated Research Affiliates of the Center); (e) fellows and "trainees" at the post-doctoral level; (f) graduate students from the participating institutions who might hold predoctoral fellowships or research assistantships at the Center.

Structure

The following organizational structure has been envisioned for the Center:



* Representing the participating New England institutions

** Representing the national scientific and technological community

*** Including the Center Coordinating Committee

**** Aided by a Fellowship Selection Committee and a Clinical Advisory Committee

Host Institution

As host institution, MIT would assume various management, fiscal and long-range planning responsibilities consistent with its own resources and the requirements of the Center. The Center's Directorate would report directly to the provost of MIT.

Institutional Advisory Board

To insure the responsiveness of the Center to the needs of the community and the participating institutions in the New England region, the latter would designate representatives to serve on an Institutional Advisory Board.

Board of Advisors for Science and Technology

The scientific and technical quality of the Center's programs would be monitored by a board appointed by the host institution with the advice of the Institutional Advisory Board. The membership of the Board of Advisors for Science and Technology would be drawn from the national community of scientists and engineers whose fields of interest and competence relate to the broad objectives of the Center.

Directorate

In consultation with the Institutional Advisory Board, the host institution would appoint the Center's Directorate. The Directorate, acting with the advice of a Center Coordinating Committee, would have responsibility for the day-to-day management and operation of the Center.

SOME CONSIDERATIONS PERTINENT TO REGIONAL RESOURCE CENTERS

The establishment of viable regional resource centers at the interface of the biomedical and computer sciences is bound to be accompanied by a host of problems -- some generic to this type of endeavor, others arising from particular prevailing circumstances. Insight into some of these problems may be provided by our experiences in trying to create a New England Center for Computer Technology and Research in the Biomedical Sciences. Bearing in mind the customary caveats about extrapolations based on a highly specialized set of conditions, it may nonetheless be pertinent to make some comments about problems likely to be encountered in establishing other regional resource centers.

A. Problems of Interdisciplinary Research

In a general sense, the members of any technical community may be located in a multi-dimensional space some of whose principal axes are:

(1) "Pure" research - "Applied" research

(2) Theoretical - Experimental

(3) Physical - Biological

(4) Individual research - Group research

(5) Research - Development

The neighborhood in this space will determine to a surprising extent the aspirations, prestige symbols, mode of publication, working habits and working rules, salary ranges, need for academic affiliation, etc., of the individuals so located. When, as in our case, the contemplated resource center

would require the services of groups located in quite different neighborhoods of the space -- so called inter-disciplinary research -- then the differences listed above become crucially important. As an example, we contrast engineers (particularly computer engineers) with experimental (non-clinical) biologists in terms of these variables.

The vast majority of engineers would expect to receive industrial-scale salaries, currently about 50 percent higher than those of biological scientists with comparable experience. Engineers are accustomed to working in project groups in which each member has a fairly specific assignment and a deadline. Collaboration among biologists is usually informal and the freedom to proceed as one pleases is jealously guarded. Engineers tend to publish infrequently and usually limit their documentation to progress reports and in-house memoranda. Conversely, biologists are committed to publishing their current findings in professional journals. The great majority of biologists are faculty members in an academic institution and conduct their research in conjunction with a teaching program while only a minority of engineers have academic affiliation and do any teaching. A biologist's scientific contribution and the value placed upon it is judged primarily by other biologists who can build on this contribution; an engineer's development is evaluated by users who will, in general, not be engineers and his prestige will depend on the social worth of the tool he has created.

In an attempt to minimize the effects of these different sets of values, we were led to the formulation of the following general principles.

First priority must be given to the autonomy of each Center staff member to pursue his program in his own way. The worth of an individual's program would always be evaluated informally by his professional colleagues and would be periodically reviewed by the Board of Advisors for Science and Technology.

However, each investigator must feel that he is free to follow where his curiosity and good sense lead him, so that as long as he remained within the broad constraints of the Center's programs, he would not be importuned or his position placed in jeopardy.

Clearly, this essential guarantee would have to be given another interpretation when the investigator is part of a group working on some large systems development. In that case, the group project's requirements would need to be given priority over the individual's freedom to choose what he wants to do. Whether decisions are made by an authoritarian group leader, by a democracy of the workers, or by some intermediate mechanism would depend on the nature of the specific project and the personalities of the group members. In every group, the group leader needs to be given responsibility and hence authority to make decisions.

The preceding paragraph highlights the essential lack of symmetry between the technological and the biological parts of the Center. As an example, if a "circuits" man were needed, the Associate Director for Computer Technology could attempt to recruit one, making the specific project assignment a condition of employment. If the project is one which enjoys high prestige, he should have relatively little trouble finding a suitable candidate. The Associate Director for Biology might wish to find a neuro-embryologist. It is almost inconceivable that he could induce an established biologist to accept a job in which the specific direction of his research would be prescribed. On the other hand, if a particular biological group has sufficiently high prestige, it would in all likelihood be able to find a post-doctoral fellow to undertake a particular assignment as part of his education. No comparable practice is found among engineers, regardless of their level of formal training. Except for those who choose academic appointments, engineers look for permanent

employment immediately upon graduation and a pool of post-doctoral fellows is as yet non-existent.

Second to the necessity for the individual's autonomy is the need to protect the integrity of project groups. For example, in the Center at the outset, there would be a number of people concerned with the explication of electrical activity in the nervous system and they could be considered the nucleus of such a project group. On the technological side we could anticipate certain other project groups -- in circuits, magnetic materials and the like. It is of great concern to the Center to be able to recruit additional personnel who would provide skills lacking in the existing staff. While the size, facilities, space, etc., of each group would be policy decisions to be determined by the Center administrative apparatus, the choice of new personnel for an already existing group should in large measure reflect the judgment of its present members.

Because of the nature of their past professional training, many of the members of the Center would have little experience in the classical disciplines of the health sciences, i.e. in bacteriology, biochemistry, physiology, medicine and the like. To make its contribution to the health-sciences, the Center would need to be embedded in a research setting with a rich and continuous source of biological and medical problems requiring the talents and competences of Center members in the physical sciences and technology.

In any branch of science or engineering, the individual worker obtains many of his ideas from discussion with others of similar professional background and interests. It would be impossible to obtain workers of high quality if their role were to be exclusively that of service or if they were to work in an environment in which their scientific specialty were sparsely represented. There is little experience to be used as a guide in deciding on the appropriate

size for a resource research center. Existing mathematical and technological groups associated with biomedical institutions are all considerably smaller than those of the planned Center. Within a university, size is of lesser importance because there are departments of mathematics, applied physics, engineering, biology, physiology, etc., to ameliorate the feeling of professional isolation. The problem of appropriate size, taken together with the breadth of the task with respect to the health sciences, suggests that the contemplated plan for the Center is a lower bound on the size of a resource of the highest quality.

B. Service Aspects

Since the development of resource centers for the health sciences is a relatively new undertaking, there exists no clear conception of the forms which working relations should take between the center and "outside" scientists. At one extreme, a resource center can be modeled on a service facility such as a computation laboratory or an instrumentation shop. In that case the center's personnel and equipment would be employed primarily in solving problems or otherwise filling the needs of scientist "customers."

At the other extreme, the model might be a university research laboratory in that the center's facilities would be devoted exclusively to problems of the intra-mural research program. Contact with other scientists would be maintained on an informal basis but with no obligation to work on problems brought in from outside.

While for certain specific tasks or in particular localities, resource centers will be formed with these models in mind, objections can be raised to the universal applicability of either extreme. So far as the service-type center is concerned there is good reason to believe that professionals of high

quality and research potential would be unwilling to devote all or even a major part of their energies to problems not of their own choosing. The difficulty encountered in trying to hire the best qualified people is well known to heads of computer and programming services, instrument development groups and similar service-oriented facilities.

The research-laboratory model will, in general, make such a center much more attractive to the appropriate specialists. Consequently, a resource center -- whatever its substantive character -- will be a repository of skills and facilities in short supply or perhaps even non-existent elsewhere. Yet the scientific community at large may have serious need to use such resources on a variety of problems, not necessarily related to those being studied by center staff members. To some extent, the very uniqueness of the center's facilities entails an obligation to consider how the needs of the general scientific community may be met. Indeed, even if no formal mechanism for dealing with such needs were established, it seems certain that requests to use the center's facilities would be made informally and pressures applied to individual staff members.

The advisory facility proposed for the Center for Computer Technology and Research in the Biomedical Sciences was intended to formalize the mode of interaction between the Center and the scientific community. The goals in mind were (1) to find a procedure to enable Center staff members to pursue their research unburdened by onerous service demands and needless incursions on their time and (2) to provide any scientist with an evaluation of the computational and instrumental requirements of his particular problem and advice on how to obtain the necessary assistance.

To these ends, the Center had planned to establish an Office for Training and Advisory Services. This Office would maintain a full-time staff primarily for administration and for preliminary consideration of problems which would be brought to it. The Office would employ a small number of programmers to expedite the solution of problems that could be treated by more-or-less standard computational techniques. To arrange computer time for such problems, liaison would be maintained with neighboring computation laboratories.

Some Center staff members would commit a fraction of their time, up to perhaps 50%, to this Office. Evaluation of incoming problems would in general be made by these individuals but, as needed, other staff members with appropriate competences would be called on for consultation.

The service role is somewhat distasteful to a number of scientists. However, many physical scientists, engineers and applied mathematicians would regard participation in such a service as part of their education and as an introduction to an area of the biomedical sciences. Furthermore, in the course of such consultations, the person being consulted may well become sufficiently interested in a biological problem to collaborate with its originator. In that event, what started as a mere service function would have led to substantial collaboration on a life science problem.

PART II

INTERIM REPORT ON THE LINC EVALUATION PROGRAM (LEP)

Introduction

The Laboratory Instrument Computer (LINC) Evaluation Program evolved out of considerations regarding the type of computing equipment that might be optimally useful in the biomedical laboratory. Of course, simple utilitarian aspects are not the only characteristics desirable in such a machine. Other considerations must also be taken into account in the design of a computer for these applications, such as the relative novelty of computer usage in the biological and medical disciplines and the traditional level of funding for such research.

A discussion of design criteria will indicate why the LINC evolved in the form that it did and will further serve to pinpoint the particular features of the machine that will be under scrutiny in the evaluation program.

Attached to this document as Appendix 1 is a description of the LINC written by W. A. Clark and C. E. Molnar which contains the following quotation:

> "In designing the LINC, the principal underlying objective has been to maximize the degree of control over the instrument by the individual researcher. Only in this way, it is felt, can the power of the computer be usefully employed without compromising scientific objectives."

This description lists the specific design objectives underlying the LINC development.

The first of these objectives refers to the physical and conceptual scale of the machine. The LINC is intended for laboratory use, whether the laboratory be a one-man or small group operation; this implies that the machine should be physically small enough and simple enough to be operated, administered, programmed and maintained by a single person should that be

necessary. It is obvious that a stored-program, general-purpose digital computer cannot be built at a cost comparable to that of a standard biomedical instrument such as an oscilloscope, though its cost might be held within the range of an electron microscope. It was therefore decided to make the cost of the LINC and its operation comparable to that of an electron microscope with appropriate accessories. It was hoped that its relatively small cost and its simplicity of operation would make it possible for machine use to be governed by scientific need instead of fiscal necessity and administrative convenience. This concept of considering the computer as a laboratory tool stands in marked contrast to the practices of most computer centers.

The second design consideration seeks to provide the experimenter with effective control of the machine from a console which immediately displays data and results for viewing and photographing. This design feature provides the experimenter with direct and immediate access to his findings; it seeks to minimize the interval between the initiation of an experimental operation and the acquisition of results.

The third design objective requires that the computer be fast enough for simple "on line" data processing and logically powerful enough to perform more complex calculations later if required.

The fourth objective pertains to the flexibility with which LINC may be integrated into a wide variety of laboratory situations. It should be capable of receiving both analog and digital inputs from such devices as amplifiers, timers, transducers, plotters, etc. while minimizing the need for complex intermediary devices. Implicit in this requirement is the concept that the LINC should be complete within itself and require none of the peripheral equipment typically associated with digital computers. All too

frequently the acquisition of a computer sets off a continuing proliferation of peripheral equipment.

The next design objective seeks to incorporate features that will facilitate training scientists, unfamiliar with digital computers, in the use of the LINC. This objective should not be taken lightly; the rate at which digital computers will find their way into biomedical laboratories will depend critically on how well biological scientists understand and are able to apply computer techniques.

The evaluation program will try to ascertain how well the LINC meets these design objectives and to determine its mechanical and electronic reliability in prolonged laboratory usage. More generally, the LEP will aim to assess whether the use of digital computers in the biomedical laboratory significantly advances the acquisition of knowledge.

Selection of Participants for the LINC Evaluation Program

Contract PH43-63-540 required the selection of a scientific advisory committee charged with choosing participants for the LEP; this advisory committee, which was called the LINC Evaluation Board (LEB), was to monitor the progress of the program and to make a final evaluation. (See Appendix 2 for list of LEB members.) The membership of the LEB was selected with the aid of administrators in NIH and NASA according to two criteria: (1) Scientific competence in the major disciplines of the basic health-related and engineering sciences and (2) equitable geographic distribution.

Following the selection of the LEB, an announcement was placed in the February 22, 1963 issue of Science (see Appendix 3). At the same time a similar notice was sent to the chairmen of university Biology and Zoology

departments and to deans of all medical schools. A deadline of 15 March 1963 was specified for receipt of inquiries concerning participation in this program. A letter (see Appendix 4) was sent to the 142 investigators who made inquiries. This letter invited submission of a proposal to the LEB before April 12, 1963. Seventy-one such proposals were received before the deadline; a list of these persons and the disciplines they represent may be found in Appendix 5.

On the 19th and 20th of April 1963 the LINC Evaluation Board convened to select participants in this program. The Board first discussed and defined criteria for the selection of participants: priority was to be given to investigators whose basic scientific problems showed little possibility of solution without the application of computational techniques; proposals that fell within that category were further examined to determine whether their needs matched the unique capabilities of the LINC.

Every attempt was made to secure a fair representation with respect to both discipline and geography. An examination of those selected will show how well these objectives were achieved. Unfortunately no inquiries or proposals were received from the west central or mountain states, but otherwise the inquiries and proposals submitted reflect population and institutional density. Those selected and four alternates were notified by the 22nd of April 1963 (see Appendix 6).

The LEB next turned its attention to the program as proposed by the staff in the Center Development Office (CDO). To the latter it seemed that the most efficient procedure for providing appropriate background training would be to have each participant assemble his own machine from prefabricated

components. During each of two so-called assembly phases, July 1-25 and August 5-30, the participants were to be given lectures on programming, operation, and maintenance of LINC. They would then proceed to assemble their own LINC under the guidance of CDO personnel and then begin to write special programs applicable to their own investigations. This proposed format was accepted by the LEB.

LINC Development

Before, during and after the deliberations of the LEB, the LINC development group, under W. A. Clark, was working day and night to insure that the LINC components would be delivered at appropriate dates for use in the two assembly phases. Those involved in this task included W. A. Clark, C. E. Molnar, of the Air Force Cambridge Research Laboratory, S. Ornstein, W. Simon, Mary Allen Wilkes, and M. Stucki with the able assistance of N. Kinch and the CDO technicians and secretarial staff. In view of the small size of this group and the tight schedule, the decision was made to subcontract for the manufacture of LINC components whenever possible.

The LINC prototype frame was delivered in the middle of June and immediately subjected to a rigorous testing procedure to ascertain what modifications might be required on the remaining machines.

Assembly Phases

The individuals involved in each of the two assembly phases are listed in Appendix 6. An initial estimate of the effort necessary for successful mastery of the material to be presented in each assembly phase showed that

a minimum work week of 5 1/2 days, each consisting of about 10 working hours, would be required. The schedule was planned as follows: 1) 8 days of lectures on the fundamentals of digital computers including programming, logic, and operating procedures, and 2) 14 days for assembly, maintenance, and operation with special emphasis on programming applicable to the participants' laboratory problems.

This schedule could not be adhered to with the first assembly group. As shown by the test procedure on the LINC prototype, the number of modifications required on the main frames resulted in a 4-day delay in their delivery. As things actually transpired, however, this simply called for a reallocation of time rather than an absolute loss of time. A greater difficulty arose from the fact that a number of modifications had to be made on the machines after their arrival at CDO. As the assembly of the machines progressed, various difficulties arose which had to be remedied on the spot, resulting in a "loss" of perhaps four or five days. Thereby, the first assembly group gained substantial though unplanned trouble-shooting experience.

The second assembly group maintained almost to the day the initially envisioned schedule. This is not to say that everything went completely smoothly, but certainly there was not the air of breathlessness that characterized the first group with respect to finishing on time. In both cases the assembled machines were shipped out by van on schedule (see Appendix 7).

Both assembly groups seemed to have gained the requisite understanding of the LINC. Thus, the amount of time assigned to the assembly phase seems to have been approximately correct.

Appendix 8 contains the teaching materials used in the course. The tutorial sessions during the early part of each assembly phase were presented

by Mr. I. Thomae of CDO; Messrs. Molnar, Ornstein, Simon, Clark, and Miss Wilkes also presented lectures covering special aspects of machine organization and use.

Continuation of the LINC Evaluation Program

Present plans call for the continuation of this program until March 1965. Field visits to various laboratories by CDO staff should assist the investigators in LINC usage. Such visits have already been made: Mr. Clark has been in Miami, at Johns Hopkins University and at Stanford. Mr. Thomae has visited the University of Wisconsin, Washington University, Johns Hopkins University and Brown University. All of the remaining laboratories are to be visited before the first of December.

Two joint meetings of the LEB and participants in the LEP are planned: the first around May 1964 and the second around March 1965. During these meetings each participant is to report on the performance of his LINC and to submit a supporting document covering: 1) his experience with machine performance and reliability, 2) adequacy or shortcomings of LINC with respect to his specific problems and 3) detailed recommendations, if any, for modifications or changes in machine design.

The Future of the LINC

The contract stated that LINC was to be developed to a level where manufacture became possible. In the opinion of the designers, LINC has reached this stage. To make manufacturing possible, comprehensive description of the machine needs to be disseminated to interested manufacturers and to this end a meeting is being contemplated within the near future. Because

of the marked pedagogical value of assembling the machine from prefabricated components it is hoped that at least one manufacturer will undertake to provide the components for a LINC in kit form. Such a kit would also have the additional advantage of a more attractive price.

Fiscal Aspects of the LINC Evaluation Program

In Appendix 9 costs of the LINC program under Contract PH43-63-540 are shown. The cost of each machine is approximately 17% higher than initially estimated. At least part of this additional cost must be attributed to the very tight schedule under which this program was undertaken and should not be reflected in the final cost of the machine if manufactured without comparable time pressure. It is worth noting, however, that the cost per machine still remains comparable to that of an electron microscope with accessories.

The nature of the LINC Evaluation Program makes the foregoing an interim progress report on a continuing program. Further reports will be forthcoming particularly following the joint meetings of the LEB and the participants in the program.

Appendix 1

THE LINC

A Preliminary Description of the
Laboratory Instrument Computer

The design of the instrument to be described reflects some of the experience gained in several years of collaboration by members of the Digital Computer Group of MIT's Lincoln Laboratory* and the Communications Biophysics Group of MIT's Research Laboratory of Electronics**. The prototype on which the present informal description is based was built at the Lincoln Laboratory with the aid of members of the Computer & Mathematical Sciences Laboratory of the Air Force Cambridge Research Laboratories. Responsibility for the further evolution of the LINC now resides, under NIH sponsorship***, in the MIT Center Development Office for Computer Technology in the Biomedical Sciences. The program is now in the final development phase, during which a number of instruments will be made available for evaluation and use in biomedical laboratories.

* Operated with the support of the U.S. Army, Navy, and Air Force.

** Supported in part by the U. S. Army Signal Corps, the Air Force Office of Scientific Research, and the Office of Naval Research; in part by the National Science Foundation; and in part by the National Institutes of Health.

*** Under contract PH43-63-540 with the Division of Research Facilities and Resources of the National Institutes of Health, in cooperation with the Bio-Sciences Office of the National Aeronautics and Space Administration.

THE LINC

A Preliminary Description of the

Laboratory Instrument Computer

by

W. A. Clark and C. E. Molnar

March 14, 1963

INTRODUCTION

There has been a growing recognition in recent years of the value of digital techniques for controlling apparatus and for handling experimental data within the laboratory. Several kinds of special-purpose digital devices, some of them capable of simple fixed calculations, have been developed by various workers and applied to problems of averaging, correlation, signal generation, event recording, etc. For the most part, each of these specially-designed devices is capable of only a narrow range of tasks and lacks the versatility desirable in a general laboratory tool.

The so-called "general purpose" digital computer is potentially the most versatile digital device by virtue of its ability to execute complex programs of internally stored instructions. However, it has been either too large and inaccessible or too slow for useful work in a laboratory environment. Furthermore, the technical development of the general purpose machine has tended to emphasize those design features which exploit its ability to solve well-defined mathematical problems in symbolic terms, with a consequent de-emphasis of features useful in processing data in less highly stylized and rigid forms.

This deficiency of traditional computer design is particularly apparent in the context of the biological or medical research program. Here the problems of achieving adequate operational flexibility, handling large quantities of data (often in the form of electrical signals) and having to deal with incompletely developed mathematical procedures all underline the need for more appropriate computer systems. The LINC is an attempt to satisfy this need.

DESIGN OBJECTIVES

In designing the LINC, the principal underlying objective has been to maximize the degree of control over the instrument by the individual researcher. Only in this way, it is felt, can the power of the computer be usefully employed without compromising scientific objectives. In particular, the goal of the development has been a machine which:

1. is small enough in scale so that the individual research worker or small laboratory group can assume complete responsibility for all aspects of administration, operation, programming, and maintenance;

2. provides direct, simple, effective means whereby the experimenter can control the machine from its console, with immediate displays of data and results for viewing or photographing;

3. is fast enough for simple data processing "on line" while the experiment is in progress, and logically

powerful enough to permit later, more complex calculations if required;

4. is flexible enough in physical arrangement and electrical characteristics to permit convenient interconnection with a variety of other laboratory apparatus, both analog and digital, such as amplifiers, timers, transducers, plotters, special digital equipment, etc., while minimizing the need for complex intermediating devices;

5. includes features of design which facilitate the training of persons unfamiliar with the use of digital computers.

The LINC design represents a reasonable balance among the conflicting requirements set by these objectives; the success of the design can be evaluated only by using the computer in a wide variety of laboratory situations.

DESCRIPTION

The LINC is a small stored-program digital computer which uses transistor circuitry and a random-access ferrite-core memory. The speed of the computer is fixed by the length of time required to read information from or store information into one of the 1024 (expandable to 2048) twelve-bit memory locations. Most of the LINC's instructions require from one to four memory-cycle times of eight microseconds each for execution. The instructions available may be grouped as follows:

1. Arithmetic instructions which perform addition, multiplication, counting, etc.

2. Logic instructions which perform simple logical manipulations.

3. Data transfer instructions which move information from one computer location to another.

4. Indexing instructions which provide a convenient means of referring to tables.

5. Input-output instructions which transfer information to and from external equipment.

6. Magnetic tape instructions which control various digital magnetic tape operations.

7. Control instructions that determine which of alternative sets of instructions are to be executed according to various criteria.

The LINC consists of four independent console modules connected (through easily detachable thirty-foot cables) to a cabinet containing the electronics and power supply (Fig 1). One console module houses most of the controls used in operating the computer as well as indicator lights. A second module contains terminals used to connect the LINC to other laboratory equipment. The remaining two modules house a display oscilloscope and a pair of specially designed magnetic tape transports which form an integral part of the LINC. The four console modules are mechanically separate and may be stacked and rearranged in any desired configuration. The front panel of each console module may be removed from its box and mounted in a standard rack with other laboratory equipment (Fig 2).

Programs are initially typed on a simple keyboard, stored directly in the LINC's core memory and simultaneously displayed on the oscilloscope (Figs 3 and 4). They may be typed either as octal numbers or, aided by a simple assembly program, in a symbolic form which uses three letter mnemonics to represent instructions. Several features which aid in the process of "debugging" programs are incorporated in the LINC. The computer can be set to stop whenever a chosen memory location is referred to, or a program may be executed slowly, one instruction at a time, while its actions are observed by means of the indicator lights on the control console. Once a program has been written and corrected, it may be stored on magnetic tape and conveniently retrieved for use at a later time.

A wide variety of both digital and analog data and control paths for connecting the LINC to other laboratory equipment has been provided. There are sixteen analog input channels connected to an internal analog-to-digital conversion device which translates an input voltage on any of these channels into an eight-bit binary number. Up to 25,000 conversions, under control of the computer program, can be made per second. A simple program will display data converted in this way on the oscilloscope (Fig 5). Eight of these analog input channels are normally connected to potentiometers whose knobs can be used as manual controls or parameter inputs (Fig 6). Two sets of twelve-bit digital input terminals may be connected to other digital equipment such as counters, timers, special encoders, or other devices which produce signals representable in parallel binary form. These digital inputs may be accepted at peak rates up to about 40,000 twelve-bit numbers per second. Another set of inputs to the LINC provides a convenient means for the computer to sense external events and synchronize itself with external equipment.

The principal output of the LINC is the oscilloscope display which may be viewed directly or photographed (Fig 7 a-f). Displays of programs, data, results of calculations, etc., can be generated point-by-point in graphical or symbolic form at rates of 10,000 to 20,000 points per second in typical programs. Letters or digits are generated and displayed by a special instruction at a rate of 4000 characters per second. There are two distinct display channels which may be connected either to the display scope mounted in the console module or else to remote standard oscilloscopes with special plug-in units. LINC provides each display scope with a pair of deflection signals which position the spot with a precision of one part in 256 along each coordinate and an intensification signal to brighten the selected location on the scope face.

The magnetic tape system was specially developed to form an integral part of the LINC system (Fig 8). It can be used to store programs, numerical data, and results on small reels of 3/4 inch wide tape which contain 512 consecutively numbered blocks, each capable of storing 256 twelve-bit numbers. The tape moves at approximately 40 inches per second in either direction and information is recorded at a track density of about 500 bits per inch. A block occupies 2 inches of tape and can therefore be scanned in about one-twentieth of a second, although the time required to start, stop, or reverse a transport is approximately one-quarter of a second.

A single computer instruction can transfer the contents of 256 core memory registers to any desired block on either of the two tape transports, or return the information thus stored to the core memory. The operation of the magnetic tape units can also be directly controlled from the console thus providing a convenient way of changing programs (Fig 9).

Output paths from the LINC to other equipment include two channels of analog output signals which are usually used to operate the oscilloscope display but which can drive an X-Y plotter or be used for other purposes. These analog output signals have a precision of one part in 256 and may be changed at peak rates up to about 60,000 times per second. Parallel digital outputs are available over one channel which provides up to 40,000 twelve-bit numbers per second. Two forms of control outputs are available, one of which consists of six sets of relay contacts operated by the computer, and the other of sixteen digital output lines which can be pulsed individually by a special computer instruction. These features make it relatively simple, for example, to install a typewriter as an auxiliary device.

A box-like design has been used in the console modules to simplify mounting and physical arrangement (Fig 10). Cables are installed from the rear to connectors on the removable console panels contained within the boxes (Fig 11).

For the most part, the electronic circuits used in the LINC are standard commercially manufactured units (Fig 12). These are in the form of cards which plug into a mounting frame held in the electronics cabinet along with the power supplies. The LINC system requires about one kilowatt of standard 115 volt a.c. power.

EXAMPLES OF APPLICATIONS

It is difficult to give a comprehensive summary of the capability of the LINC in various applications. The flexibility of a stored program machine allows one to choose, from among many alternatives, a programming scheme which efficiently utilizes the resources of the computer. Central storage capacity, speed of operation, word length, and speed of magnetic tape operation are among the factors that must be taken into account in selecting a programming scheme. For any particular problem there is usually no unique "best solution." A description of a few typical problems which have been considered and tried by various investigators may give some insight into the range of the LINC's usefulness.

Averaging of Evoked Electrophysiological Responses -- Presentation of acoustic stimuli to a cat with implanted electrodes and averaging of cortical and thalamic responses were performed by the LINC. Averaged responses to series of stimuli as well as information relating to the variability of the responses were immediately displayed and also automatically stored on the LINC magnetic tape for more detailed examination at a later time.

Fourier Analysis -- A program has been written to perform a Fourier analysis of electron diffraction data from thin metal films and to resynthesize the original data from its Fourier components for verification of the analysis. The Fourier components and the resynthesized data are displayed on the oscilloscope along with the original data. The analysis and resynthesis carried to 50 harmonics takes about one minute.

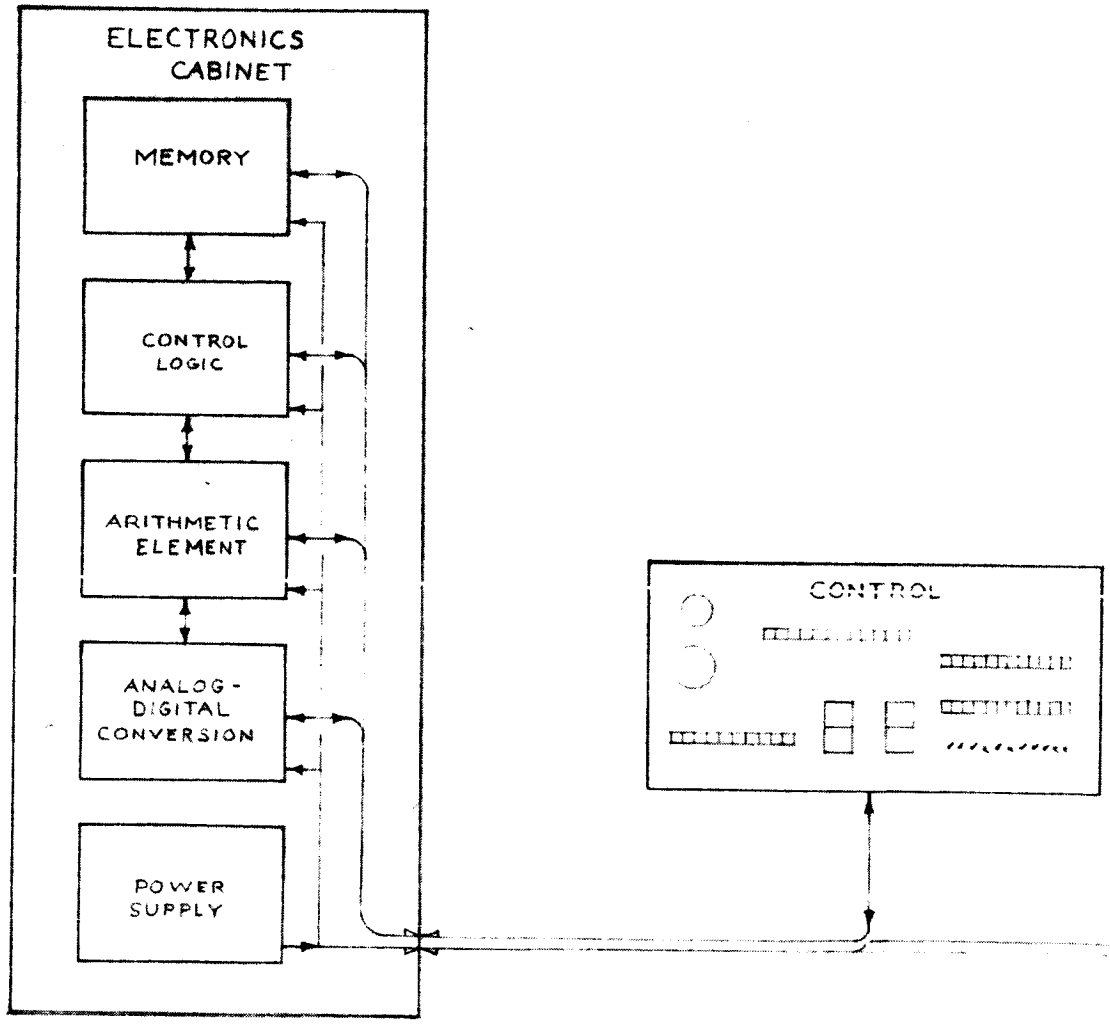
Resolving a Sum of Decaying Exponentials -- A problem in compartmental analysis required a program to resolve a sum of decaying exponential signals into its individual components. This was done by displaying on the oscilloscope the logarithm of the waveform being analyzed and fitting a straight line to portions of the resulting curve. With the parameter knobs, the experimenter adjusted the slope and position of a straight line also displayed on the oscilloscope to get the best fit to the data. The component thus determined was subtracted from the original waveform and the process repeated with the remainder until all of the components were resolved.

Processing of Single-unit Data from the Nervous System -- Programs have been written to determine, from micro-electrode recordings, the times at which single neurons fired, and to calculate the distribution of intervals between successive firings. These programs can also be used to determine the distribution of firing times following the presentation of a discrete stimulus.

Cursor Program -- An experimental curve stored in the memory of the LINC can be displayed on the scope along with an adjustable cursor mark. This cursor designates a desired point on the curve and its location is controlled by a parameter knob. The amplitude of the point under the cursor is displayed numerically on the scope.

Arterial Shock Wave Measurements -- A LINC program has been written to make comparative hydrodynamic measurements in the ventricular cerebro-spinal system in order to determine the dissipation and attenuation factors in shock waves attributable to the arterial pulse. The LINC program was designed to work directly with amplified signals from strain-gauges.

In-phase Triggering of Stimuli from EEG Alpha Wave -- Simple criteria have been applied to portions of EEG signals to identify and mark the occurrence of rhythmic bursts of alpha activity, and to trigger stimuli which are phase-related to the alpha wave.



1024 (EXPANDABLE TO 2048) 12-BIT REGISTERS OF FERRITE-CORE MEMORY, 8 μSEC CYCLE TIME

STORED-PROGRAM LOGIC, 50,000 INSTRUCTIONS PER SECOND

12-BIT BINARY ARITHMETIC

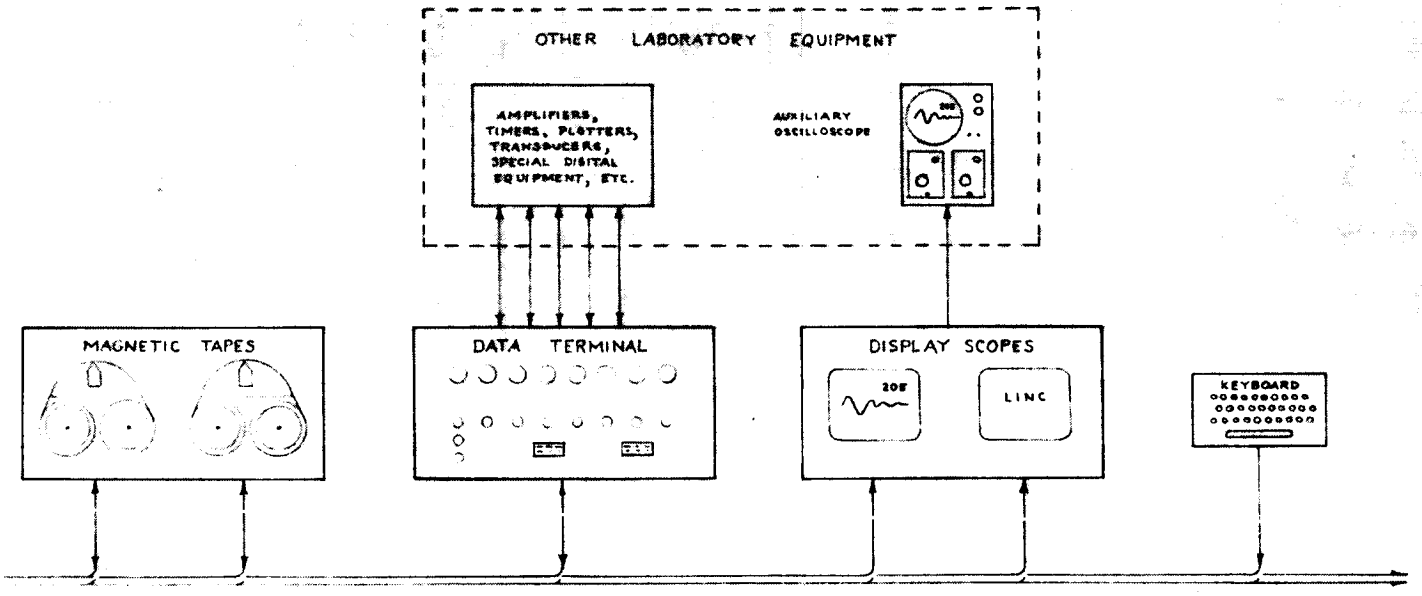
25,000 ANALOG-DIGITAL CONVERSIONS PER SECOND; PRECISION 1 PART IN 256

POWER CONSUMPTION ABOUT ONE KILOWATT STANDARD 110 VOLT AC

REGISTER INDICATOR LIGHTS, PUSH BUTTONS, SWITCHES

MANUAL CONTROL OF ALL MACHINE FUNCTIONS, WITH PROVISIONS FOR EXECUTING INSTRUCTIONS ONE STEP AT A TIME.

PROVISIONS FOR SETTING AND SCANNING MEMORY CONTENTS.



BLOCK-ADDRESSED, 512 BLOCKS PER TAPE, 256 12-BIT NUMBERS PER BLOCK.

SINGLE-INSTRUCTION BLOCK SEARCH AND TRANSFER AT 40 INCHES PER SECOND. ONE BLOCK OCCUPIES .2 INCHES OF TAPE.

START, STOP, OR REVERSE ACTIONS REQUIRE ABOUT 1/4 SECOND.

16 ANALOG INPUT CHANNELS (8 NORMALLY USED FOR PARAMETER KNOBS) INPUT RANGE -1 VOLT TO +1 VOLT

TWO SETS OF 12-BIT DIGITAL INPUT TERMINALS, 40,000 12-BIT NUMBER-TRANSFERS PER SECOND

16 TERMINALS FOR SIGNALS USED IN SYCHRONIZING LINC WITH EXTERNAL EQUIPMENT

12-BIT DIGITAL OUTPUT CHANNEL, 40,000 12-BIT NUMBER TRANSFERS PER SECOND

6 SET OF RELAY CONTACTS OPERATED BY LINC INSTRUCTIONS

16 DIGITAL OUTPUT LINES WHICH CAN BE INDIVIDUALLY PULSED BY INSTRUCTION

2 DISPLAY CHANNELS, POINT-BY-POINT OPERATION AT 80,000 POINTS PER SECOND

POSITION PRECISION 1 PART IN 256 ALONG EACH COORDINATE

SPECIAL INSTRUCTION FOR DISPLAY OF SIMPLE CHARACTERS AT 4000 PER SECOND

OTHER OSCILLOSCOPES CAN BE OPERATED IN PARALLEL

ELECTRICALLY ACTUATED 96 KEYS, 6-BIT CODE FOR EACH KEY

CHANGE LETTER	APP'D BY	DATE	CHANGES

LINC

GENERAL CHARACTERISTICS

ENG. WILKES + CLARK

DATE 3/13/63

246

CH.

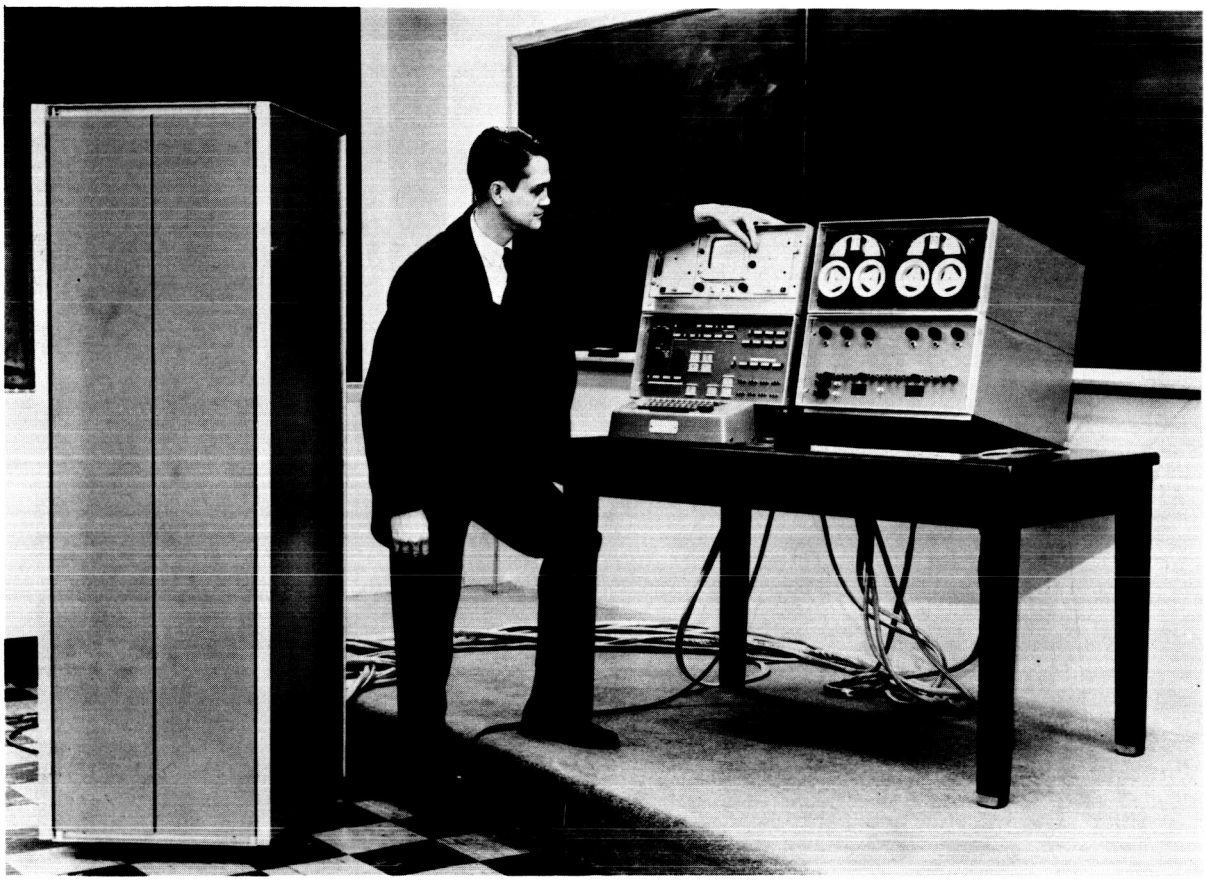


Fig 1 The LINC (Laboratory Instrument Computer) is a small stored-program digital computer designed to accept analog as well as digital inputs directly from experiments, to process data immediately, and to provide signals for the control of experimental equipment. The LINC system comprises five physically distinct subassemblies which include four console modules connected by separate cables to a remote cabinet containing the electronics and power supplies. The control module contains indicator lights, pushbuttons, and switches used in operating the LINC. A second module provides for display oscilloscopes, while a third module holds two magnetic tape transports of special design. The last module is provided with sockets, jacks, and terminals for interconnecting the LINC and other laboratory equipment. The photograph shows the version demonstrated on March 27, 1962, at the M.I.T. Lincoln Laboratory.

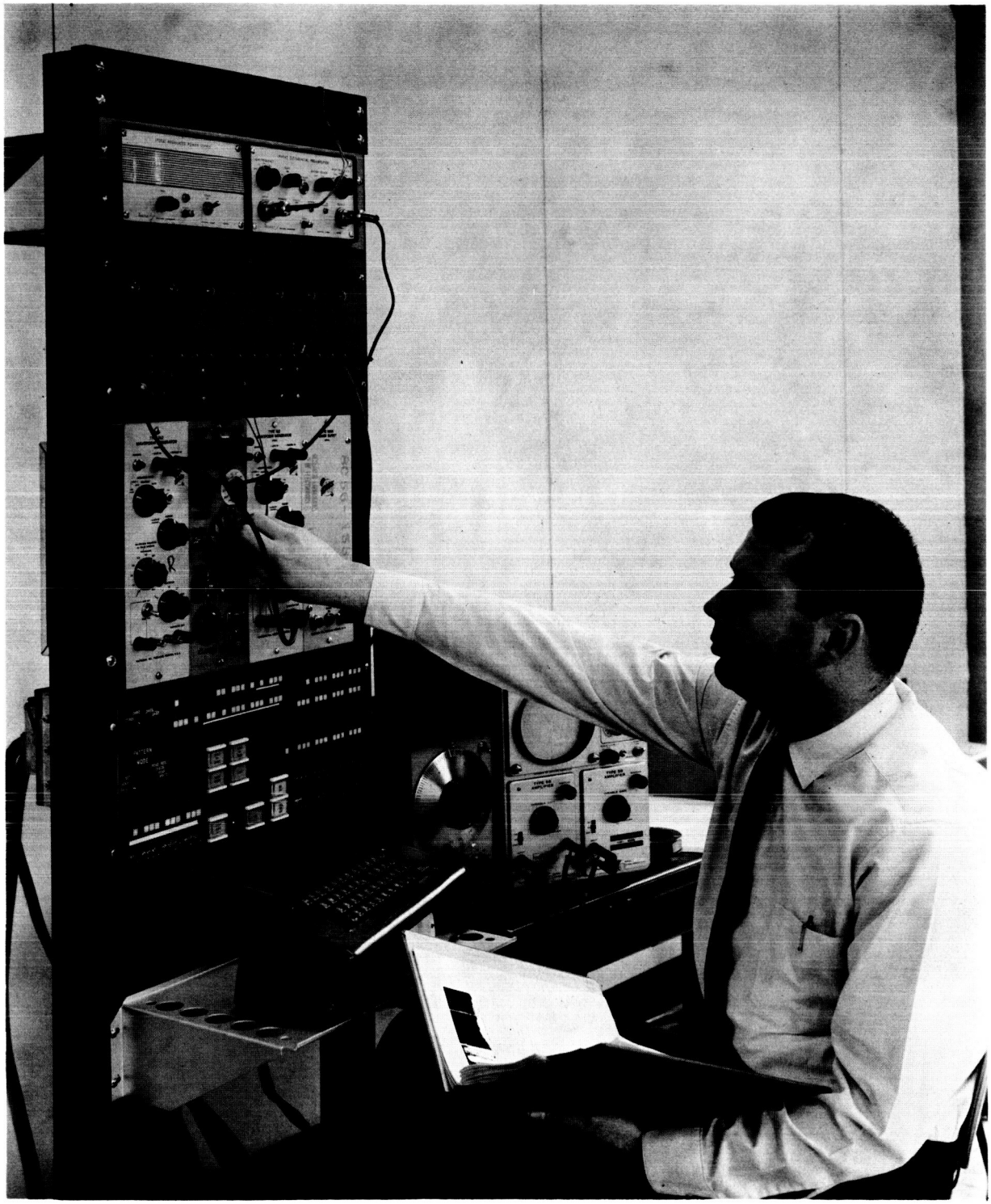


Fig. 2 Two LINC console panels mounted in a standard rack with other laboratory equipment in an arrangement convenient for simple electrophysiological experiments. The oscilloscope on the table can substitute for the console module scopes shown in other figures. The remaining panels are operating in another part of the room.

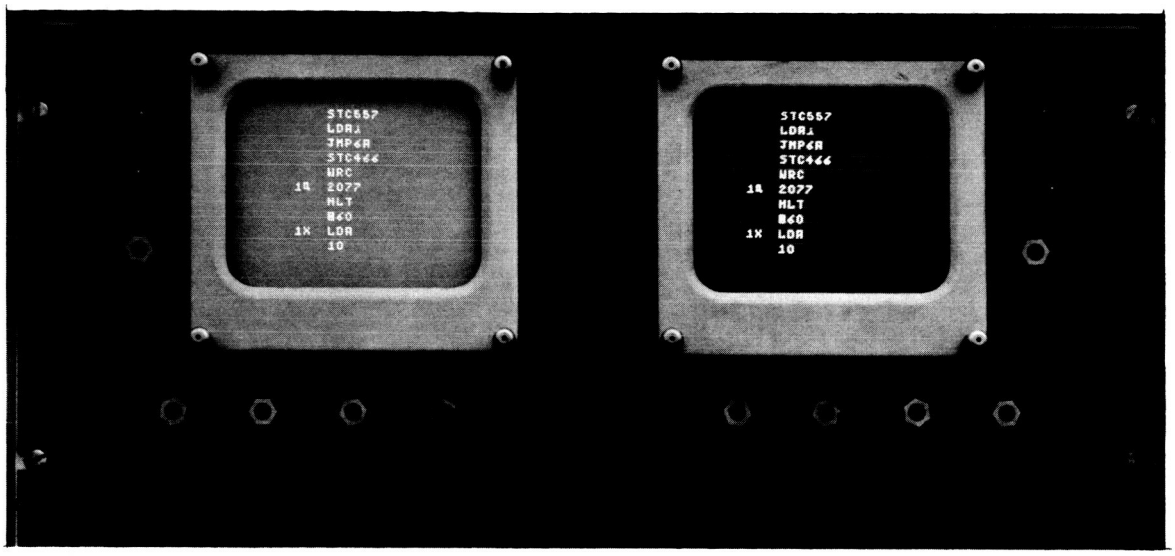


Fig 3 The oscilloscope is a primary means of communication between LINC and user. A typical instruction manuscript which has just been typed on the keyboard is displayed on both scopes.

Fig 4 Information typed on the keyboard and displayed on the scope is simultaneously stored in the central memory. From here it may be transferred to magnetic tape and recovered whenever desired.

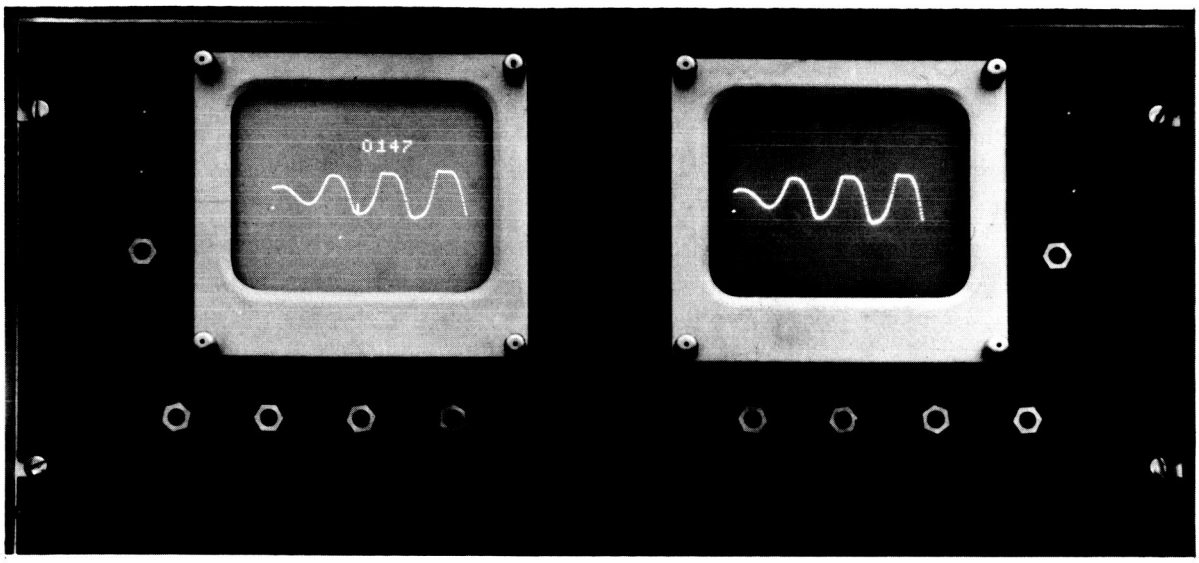


Fig 5 The oscilloscope is operated point by point and can display experimental or calculated curves as well as characters. Note that the displays on the two scopes need not be identical.

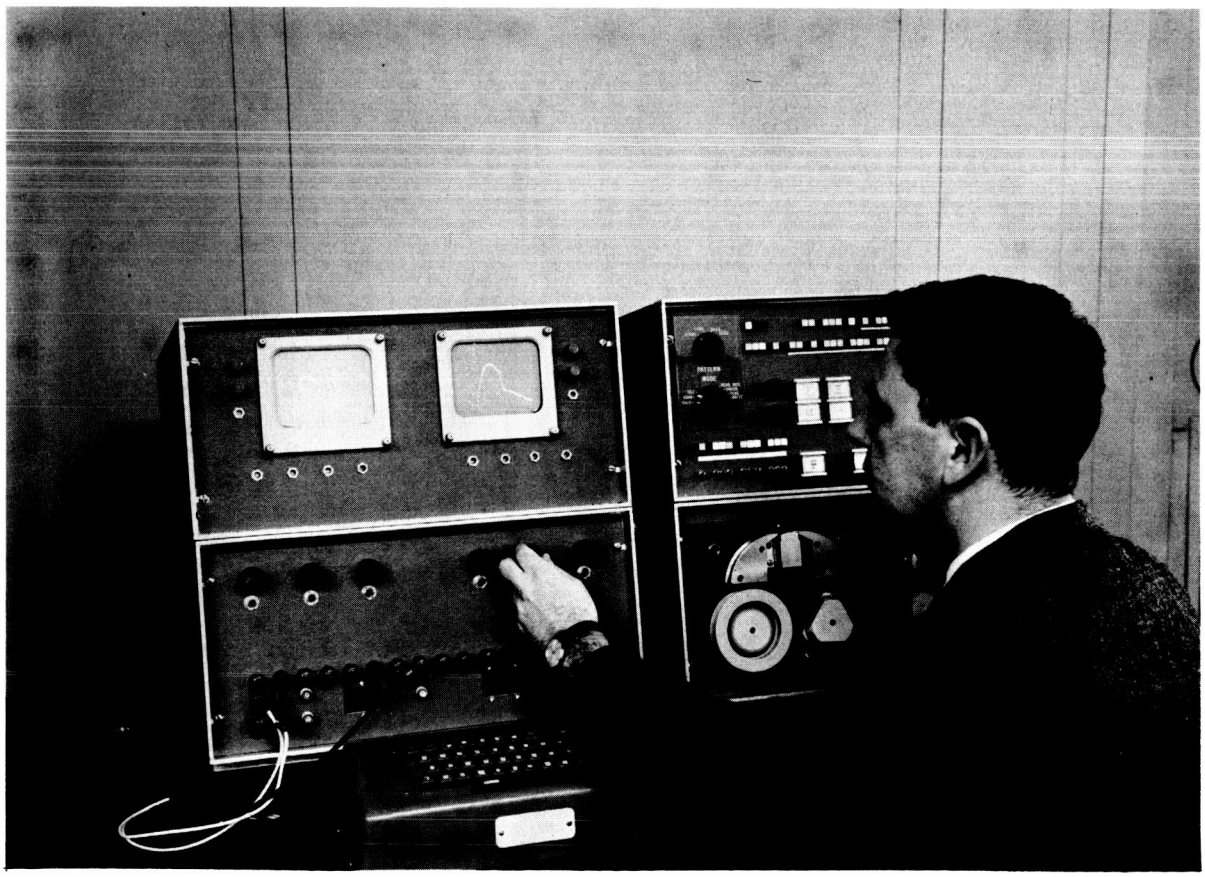
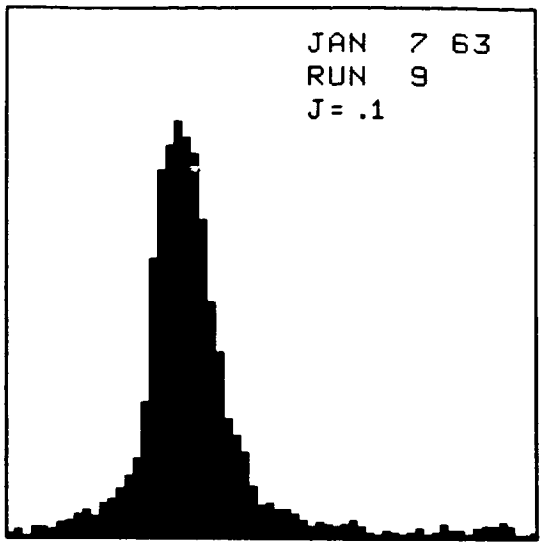
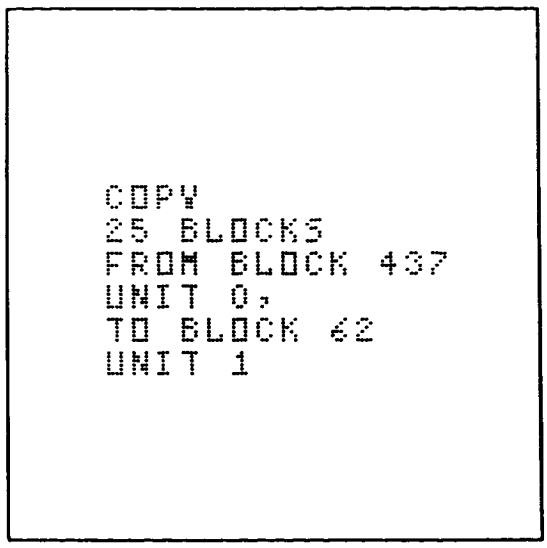


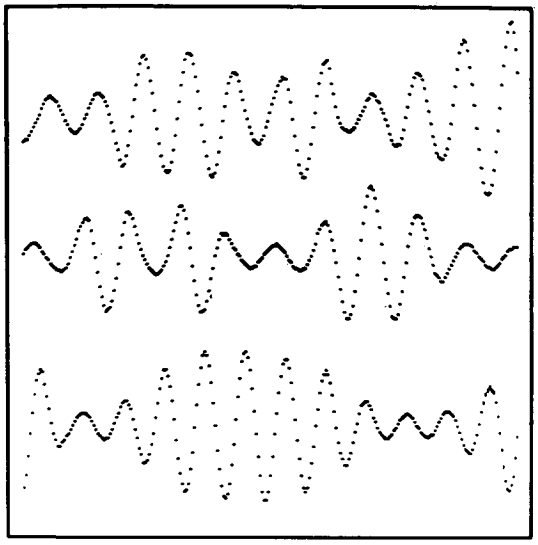
Fig 6 The knob is being used to adjust the fit of a straight line to a curve derived from a signal connected to one of the LINC's eight analog input channels.



a



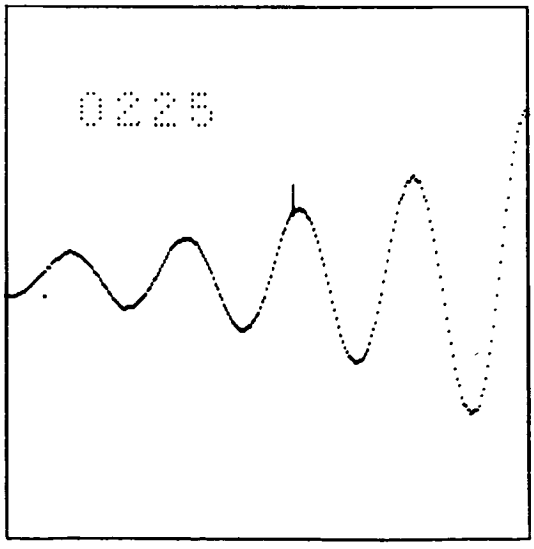
d



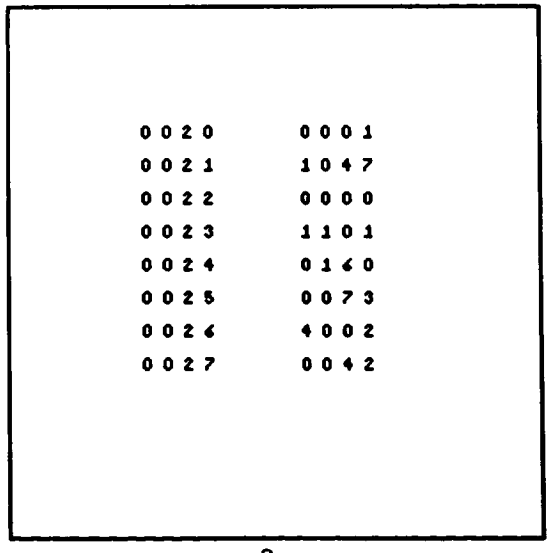
b

```
 1 [TAPE COPY          35      JMP 2A
 2       LDA I         36      XSK I 2
 3       1776          37      JMP 1B+2
 4       STC 1B        40      ADD 1B
 5       ADD 1A-3   41 •1C BCL I
 6       STA           42      6000
 7       3B+4          43      SRE I
10       STC 3A+5      44      776
11       SET I 3       45      JMP 1C+6
12       -3            46      JMP 1F
13       SET I 5       47      SRE I
14       1T-1          50      777
15 •1A LDA I           51      JMP 1A
16       2             52 •1D LDA
17       SET I 2       53      3
20       -400          54      SMD I
21       ADM           55      7400
22       1B            56      JMP 1E
23       BCL I         57      SMD I
24       1777          60      7500
25       ROR I 2       61      CLR
26       ADA I         62      ADD 1B-3
27       -1            63      STA
30       STC 1         64      3A+5
31       RDC #         65      STC 3B+4
32 •1B                 66      JMP 3A
33       CLR           67 •1E RNB #
34       SRE I 1       70      0
```

e



c



f

Fig 7 Displays generated by typical LINC programs. Photographs were made by a Polaroid camera on the oscilloscope shown in Fig 2 and reversed to improve legibility.

Fig 8 To use the magnetic tape unit, a loaded reel is snapped onto the transport and the tape is drawn over a simple open guide to a take-up reel. Pushbuttons facilitate loading and unloading.

Fig 9 The keyboard can be used to change the values of experimental parameters or to modify operating programs. Records of such changes can be stored on magnetic tape.

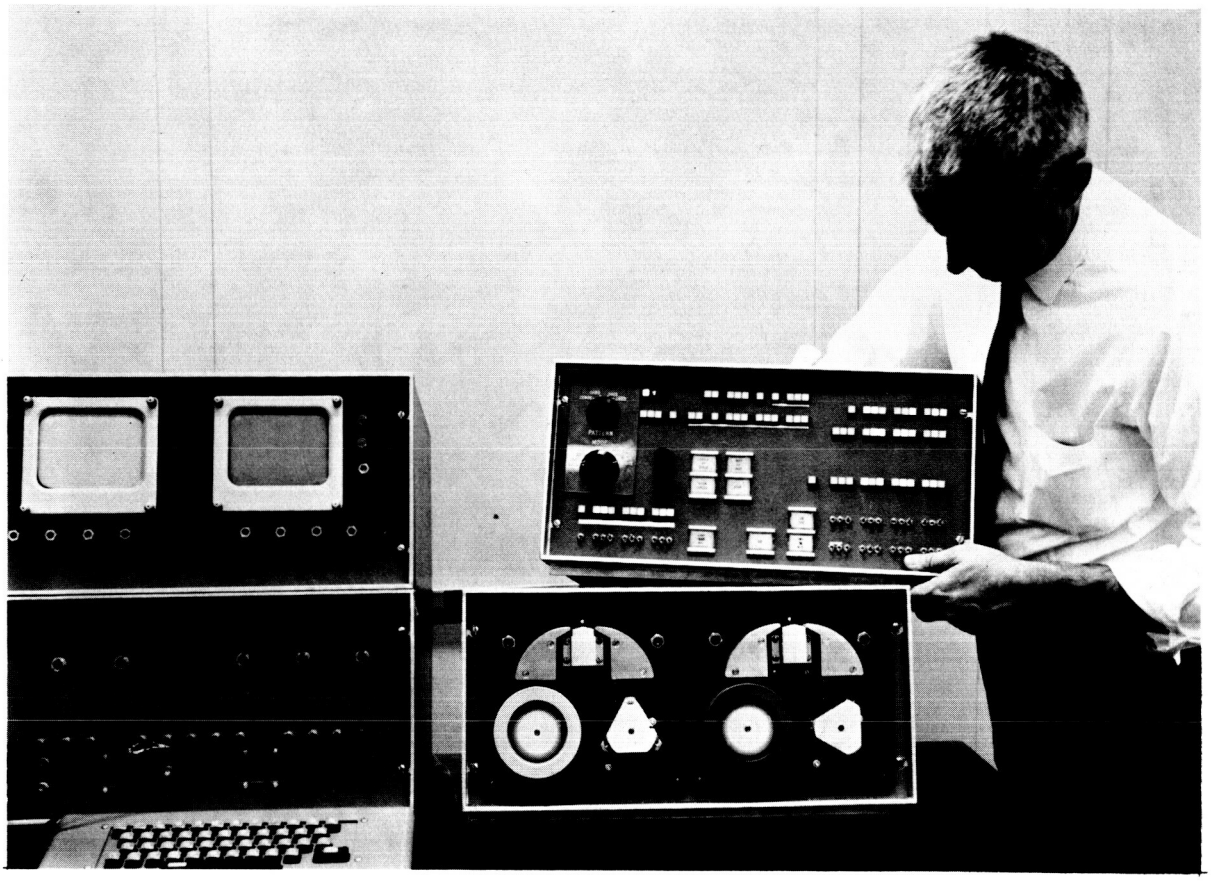


Fig 10 The control module being stacked on top of the magnetic tape module. Cables connecting console panels to the electronics cabinet are plugged in from the rear.

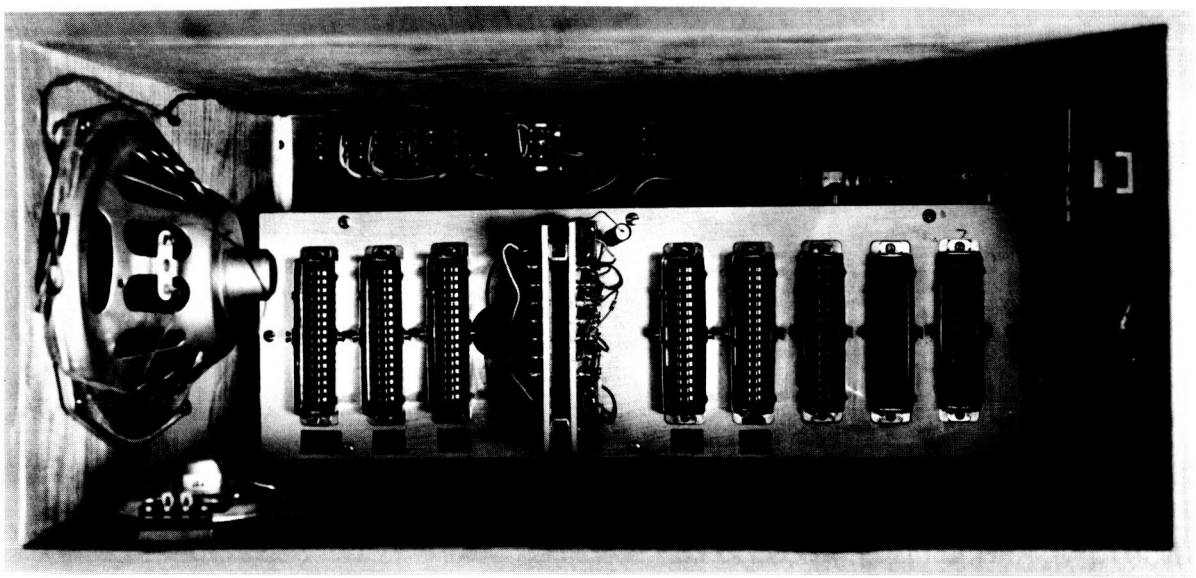


Fig 11 Rear view of the control module showing cable connectors on the back of the control panel. Loudspeakers provide audible indication of computer activity.

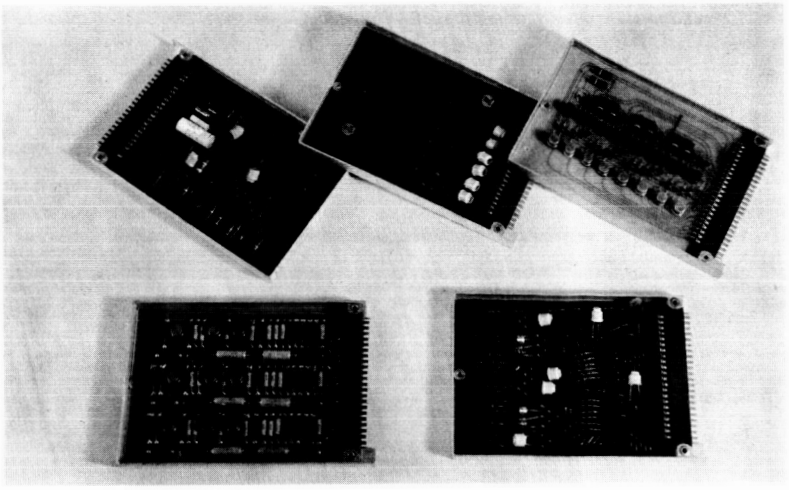


Fig 12 Typical transistor circuit plug-in units. About 275 plug-in units are used in the LINC.

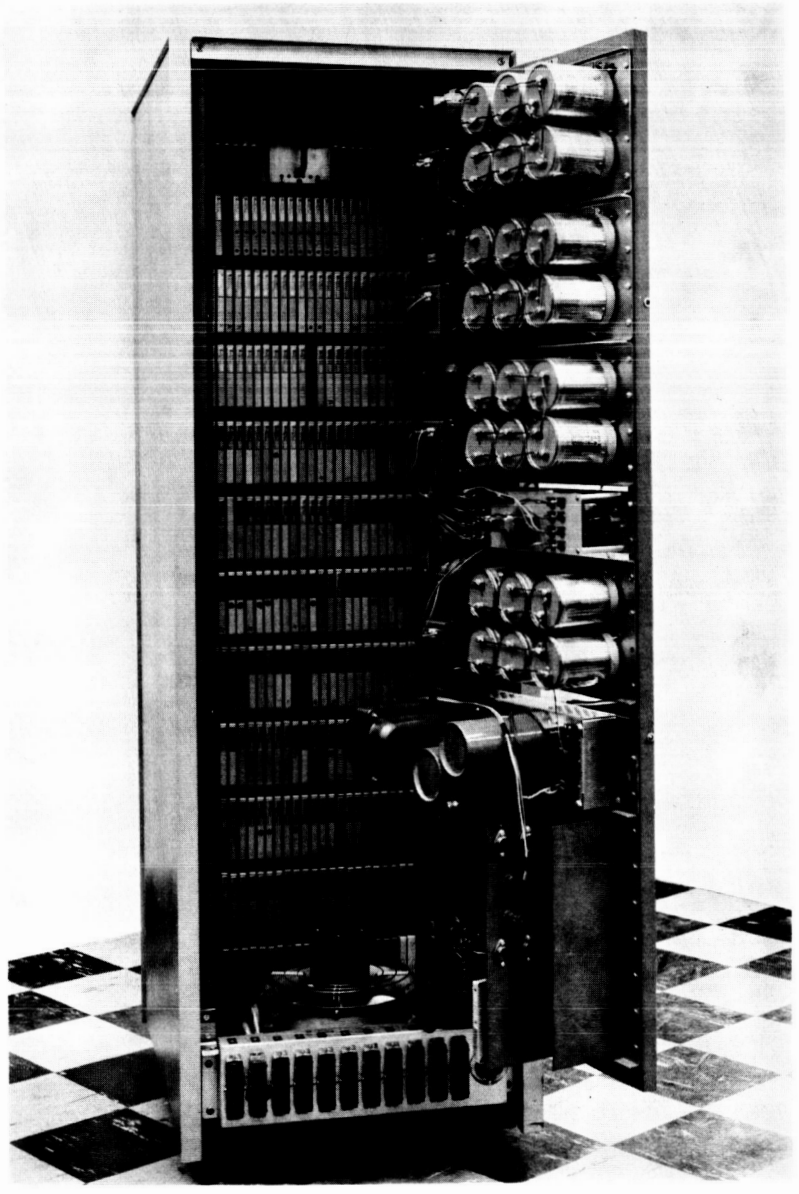


Fig 13 The electronics cabinet opened to show plug-in units and the power supply. The LINC system uses about one kilowatt of standard 115 volt a.c. power.

Appendix 2

LINC EVALUATION BOARD

Members:

Dr. Mary A. B. Brazier	University of California at Los Angeles
Dr. Jerome R. Cox, Jr.	Central Institute for the Deaf
Dr. Everett R. Dempster	University of California
Dr. Josiah Macy, Jr.	Albert Einstein College of Medicine
Dr. Hubert V. Pipberger	Veterans Administration Hospital, Washington
Dr. Murray Rosenberg	Rockefeller Institute
Mr. Harold W. Shipton	State University of Iowa

Consultants:

Dr. J. H. U. Brown	Division of Research Facilities and Resources, NIH
Dr. George Jacobs	Bio-Science Office, NASA
Dr. Herbert B. Pahl	Division of Research Facilities and Resources, NIH
Mr. Philip Sapir	National Institute of Mental Health, NIH

Executive Secretary
Dr. Thomas T. Sandel — Center Development Office, MIT

Appendix 3

Excerpt, Science, 22 February 1963

The National Institutes of Health invites biomedical scientists to participate in evaluating the usefulness of the Laboratory Instrument Computer (LINC), a small, stored-program digital computer developed at MIT's Lincoln Laboratory, with the participation of the Air Force Cambridge Research Laboratories. Participants will assist in assembling a LINC, and will provide an evaluation after using it in their own research for a year. Deadline for receipt of inquiries: 15 March. (T. T. Sandel, LINC Evaluation Board, 292 Main St., Cambridge 39, Massachusetts)

Appendix 4

LINC Evaluation Board
292 Main St., Cambridge 42, Mass.

March 14, 1963

To: Respondents inquiring about the LINC Evaluation Program

From: T. T. Sandel
Executive Secretary, LINC Evaluation Board

Thank you for the interest expressed in your recent inquiry concerning the LINC Evaluation Program. In this letter and in the appended materials you should find the answers to many of the questions we thought you might have concerning participation in this program.

The LINC Evaluation Program is one of the responsibilities assigned to the Massachusetts Institute of Technology in a contract (PH43-63-540) with the National Institutes of Health, Division of Research Facilities and Resources. This program is the responsibility of the LINC Evaluation Board, composed of competent and knowledgeable workers equitably distributed among the relevant major disciplines of the basic health-related and engineering sciences.

The proposed LINC Evaluation Program consists of two essential phases. The first phase is the assembly phase. In this phase, a principal who has submitted a proposal and has been selected by the LINC Evaluation Board for participation in the program will be expected to be present in Cambridge, Massachusetts, for a one month period, either 1-26 July or 5-30 August 1963. During this period the principal will (a) actively assist in the final assembly of his machine, (b) participate in its final testing, and (c) receive intensive training in programming, operation, and maintenance procedures related to the LINC.

While such a program may sound formidable, several attenuating considerations need to be kept in mind. Final assembly is primarily a job of installing pre-wired circuit packages in a pre-wired frame and connecting the input-output equipment to the computer electronics. (For the relevant philosophy, see the statement of design objectives in the appended informal description of the LINC.) The assembly operation acquaints the participant, in some detail, with the logical organization and the basic components of the machine, a fact that should make maintenance and operating procedures simpler.

Previous experience in two NSF-sponsored summer workshops has shown that programming fundamentals can be effectively acquired in the period set aside for the assembly phase. The assembly phase will thus be a period of intense and, hopefully, rewarding effort.

During the second phase of the LINC Evaluation Program a LINC will be placed in the participant's laboratory, where it will be integrated into ongoing research in whatever way the participant feels appropriate.

Participants will, of course, remain in communication with the LINC group in Cambridge, which will serve as a clearing house for information between the various participating groups during the evaluation period. This interchange of information should include major programs written for the various LINCs, experience with aspects of operating the LINCs in various ways, and particulars of performance during the evaluation period. Participants should be prepared to submit reports or documents that the LINC Evaluation Board may require. They should further be prepared to receive field visits by members of the LINC group in Cambridge. The purpose of these visits will be to assist with unanticipated problems and to help in the evaluation of the machine.

At the end of the evaluation period, Final Reports of reasonable detail will need to be submitted by all participants to the LINC Evaluation Board.

In summary, a participant in the LINC Evaluation Program should be prepared to:

1. Personally participate in the final assembly of his assigned computer (including training of LINC programming, operation, and maintenance procedures).

2. Actively communicate with his colleagues in this undertaking through the LINC group in Cambridge acting as an information clearing house,

and

3. Provide the LINC Evaluation Board with reasonably detailed evaluative comments, including a Final Report at the termination of the Evaluation period.

The LINC Evaluation Board, whose responsibility it is to see to an equitable allocation of the machines built under this program, needs from each respondent to this letter, a proposal which, among other things, sets forth in some detail the existing research programs in progress in the laboratory and the contribution which a LINC might make to that work. These detailed proposals will constitute the major source of information on which the LINC Evaluation Board will form its judgments.

No specified form is required for the proposal, but the following information should be embodied in a short document:

1. Name and address of institution

2. Name and title of principal investigator, i.e., participant.

3. A short description of the laboratory in which the LINC will be placed, including descriptions of facilities and personnel.

4. A reasonably detailed description of present research and the projected role of the LINC in this research. N.B. The appended description of the LINC should be carefully read before writing the research proposals. The relevance or irrelevance of this particular computer to your research needs should be the prime criterion of your decision concerning participation in this program.

5. A preference for one of the two assembly periods. (Those who wish to participate should realize that there may be difficulties in scheduling. If it were possible for some of you to keep both periods open this fact would add flexibility).

6. Any other information you may feel is relevant in your particular case or which you feel would be helpful to the LINC Evaluation Board in making its decision, such as a discussion of possible relationships to other departments in your institution if this seems germane to your needs or projected plans concerning a LINC.

The terminal date for receipt of proposals is

12 April 1963

Those selected for participation in the program will be informed before

28 April 1963

In writing your proposal, you should keep in mind that many of the details ordinarily discussed in proposals are not relevant in this case. The decision which rests on your proposal is solely whether you will be chosen as a participant in the LINC Evaluation Program. The responsibility for travelling costs incurred with regard to this program resides with the National Institutes of Health. NIH has stated its position as follows:

"The National Institutes of Health, subject only to the limitations of its general policies for the support of investigators assures that each person who is selected by the LINC Evaluation Board (LEB) for this study will be contacted by the staff of the National Institutes of Health for the purpose of providing direct support in an amount to be determined in each instance but which will be sufficient to provide for: (a) the travel costs of the investigator between Cambridge, Massachusetts, and his university; (b) subsistence for the investigator during the period he is in Cambridge assembling the LINC and being trained in its use; (c) the cost of shipping and installing the completed LINC in the investigator's own laboratory.

The LINC shall remain in the investigator's laboratory during the period of its evaluation (to be determined by the LEB, but approximately 12 to 18 months duration following the installation of the LINC) unless good and sufficient reasons lead the LEB to recommend otherwise. Following the evaluation period the responsibility for the ultimate disposition of each LINC rests solely with the National Institutes of Health, each situation to be appraised on its own merits. During the evaluation period each investigator will be responsible to the LEB for the preparation and submission of such reports as deemed advisable by the Board; copies of reports will be required by the National Institutes of Health."

As you can see from the foregoing paragraphs, those chosen to participate in the program, while having the direct costs associated with the assembly, training and shipping aspects of the LINC program paid, remain responsible for the maintenance and operation of the machine. This concept is intrinsic to the whole program, and is one of the major features to be evaluated by it. Indeed, possible to use this computer as a laboratory instrument, to "administer" and maintain it with little or no more effort than other laboratory instruments.

The answer to this question and the answers to questions concerning the other design objectives of the LINC are the basic reasons for this program.

If, on the basis of the information given here, you would like to participate in this program, your proposal will receive the careful consideration of the LINC Evaluation Board.

Appendix 5

Proposals Received for Participation in the LINC Evaluation Program

Name and Degree	College or Institution and Location	Field of Interest
Allen, M. B., Ph.D.	Kaiser Foundation Research Institute Richmond, California	Biochemistry
Andree, R. V., Ph.D.	University of Oklahoma Norman, Oklahoma	Mathematics
Attinger, E. O., M.D.	Presbyterian Hospital in Philadelphia Philadelphia, Pennsylvania	Cardiovascular Physiology
Banghart, F. W., Ed.D.	University of Virginia, Charlottesville, Virginia	Biostatistics
Barker, J. N., Ph.D.	Jefferson Medical College Philadelphia, Pennsylvania	Biochemistry, Biophysics
Bickford, R. G., M.D.	Mayo Clinic Rochester, Minnesota	Neurophysiology, EEG
Blough, D. S., Ph.D.	Brown University Providence, Rhode Island	Psychology
Boneau, C. A., Ph.D.	Duke University Durham, North Carolina	Physiological Psychology
Campbell, S. L., Ph.D.	Los Angeles State College Los Angeles, California	Psychology (Learning)
Chapman, L. F., Ph.D.	UCLA, Institute for Comparative Biology Zoological Society of San Diego San Diego, California	Neurophysiology
Coulter, N. A., M.D.	The Ohio State University Columbus, Ohio	Biophysics, Cardiovascular Physiology
Daniel, R. S., Ph.D.	University of Missouri Columbia, Missouri	Neurophysiology, Neurology
Domino, E. F., M.D.	University of Michigan Ann Arbor, Michigan	Pharmacology
Dragsdorf, R. D., Ph.D.	Kansas State University Manhattan, Kansas	Physics
Foshee, D. P., Ph.D.	University of Mississippi Medical Center Jackson, Mississippi	Neurophysiology
Gasteiger, E. L., Ph.D.	New York State Veterinary College Cornell University Ithaca, New York	Biophysics
Gluck, H., Ph.D.	University Hospitals of Cleveland Cleveland, Ohio	Physiological Psychology, Pharmacology

Name and Degree	College or Institution and Location	Field of Interest
Gordon, N. B., Ph.D.	Yeshiva University New York City, New York	Psychology (Motor abilities)
Grodins, F. S., M.D.	Northwestern University Chicago, Illinois	Analytic Physiology
Henneman, E., M.D.	Harvard Medical School Boston, Massachusetts	Neurophysiology
Herrnstein, R. J., Ph.D.	Harvard University Cambridge, Massachusetts	Psychology (Operant conditioning)
Hertzler, E. C., Ph.D.	University of Michigan Dearborn, Michigan	Neurophysiology
Higgins, E. A., B.S.	Civil Aeromedical Research Institute Oklahoma City, Oklahoma	Circulatory Physiology, Biochemistry
Hind, J. E., Ph.D.	University of Wisconsin Madison, Wisconsin	Neurophysiology
Hunt, E. E., Jr. Ph.D.	Forsyth Dental Infirmary Boston, Massachusetts	Anthropology
Ingram, W. R., M.D.	State University of Iowa Iowa City, Iowa	EEG
Isaac, W., Ph.D.	Emory University Atlanta, Georgia	Physiological Psychology
Jaffe, J., M.D.	College of Physicians and Surgeons Columbia University New York City, New York	Speech Analysis, Physiological Psychology
Jardetzky, O., M.D.	Harvard Medical School Boston, Massachusetts	Biophysics
Jett, R., M.D.	University of Cincinnati College of Medicine Cincinnati, Ohio	Renal Physiology
John, E. R., Ph.D.	University of Rochester Rochester, New York	Physiological Psychology
Johnson, G. L.	The Eastern Pennsylvania Psychiatric Inst. Philadelphia, Pennsylvania	Physiological Psychology, Pharmacology
Jones, L. V., Ph.D.	University of North Carolina Chapel Hill, North Carolina	Psychology (statistics)
Karpeles, L. M., M.D.	University of Maryland School of Medicine Baltimore, Maryland	Cardiovascular Physiology, Neurophysiology

Name and Degree	College or Institution and Location	Field of Interest
Kawin, B., Ph.D.	Veterans Administration Hospital Fort Howard, Maryland	Biophysics
Kimbel, P., M.D.	Albert Einstein Medical Center Philadelphia, Pennsylvania	Pulmonary & Cardiovascular Physiology
Kittle, C. F., M.D.	University of Kansas Medical Center Kansas City, Kansas	Cardiovascular Physiology
Koella, W. P., M.D.	Worcester Foundation for Experimental Biology Shrewsbury, Massachusetts	Neurophysiology, Pharmacology
Kushner, D. S., M.D.	Chicago Medical School Chicago, Illinois	Biophysics, Biochemistry
Latimer, C. N., Ph.D.	Lederle Laboratories, Pearl River, New York	Psychopharmacology
Lederberg, J., Ph.D.	Stanford University School of Medicine Palo Alto, California	Genetics
Leichner, G. H., Ph.D.	University of Illinois Urbana, Illinois	Biophysics
Lilienfield, L. S., M.D.	Georgetown University Medical Center Washington, D.C.	Biophysics
Lilly, J. D., M.D.	Communication Research Institute of St. Thomas Miami, Florida	Neurophysiology
Lipton, M. A., M.D.	University of North Carolina Chapel Hill, North Carolina	Psychiatry, Neurophysiology
Lyons, H. A., M.D.	State University of New York Brooklyn, New York	Pulmonary Physiology
Malindzak, G. S., Ph.D.	Wake Forest College, The Bowman Gray School of Medicine Winston-Salem, North Carolina	Cardiovascular Physiology
McCandless, J. B., M.D.	Inter American University of Puerto Rico San Germán, Puerto Rico	Biostatistics
Michie, A. J., M.D.	Children's Hospital of Philadelphia Philadelphia, Pennsylvania	Urology
Mirsky, A. F., Ph.D.	Boston University School of Medicine Boston, Massachusetts	Physiological Psychology
Morrison, N.	Associated Hospital Service of New York New York City, New York	Statistics

Name and Degree	College or Institution and Location	Field of Interest
Mountcastle, V. B., M.D.	Johns Hopkins University Baltimore, Maryland	Neurophysiology
Olds, J., Ph.D.	University of Michigan Ann Arbor, Michigan	Physiological Psychology
O'Leary, J. L., M.D.	Washington University St. Louis, Missouri	Neurophysiology
Perkins, N. L., M.D.	Maine Medical Center Portland, Maine	Cardiovascular Physiology
Pinneo, L. R., Ph.D.	Tulane University Delta Regional Primate Research Center Covington, Louisiana	Physiological Psychology
Proctor, L. D., M.D.	Edsel B. Ford Institute for Medical Research Detroit, Michigan	Neurology, EEG
Randall, J. E., Ph.D.	University of Missouri Medical Center Columbia, Missouri	Cardiovascular Physiology
Reinhardt, W. E., B.S.E.E.	State University of Iowa College of Medicine Iowa City, Iowa	Biomedical Engineering
Salisbury, P. F., M.D.	St. Joseph Hospital Burbank, California	Cardiovascular Physiology
Schwan, H. P., Ph.D.	University of Pennsylvania Moore School of Electrical Engineering Philadelphia, Pennsylvania	Biomedical Engineering, Biophysics
Seligson, D., M.D.	Yale University School of Medicine New Haven, Connecticut	Biochemistry
Shepard, R. B., M.D.	University of Alabama Medical Center Birmingham, Alabama	Cardiovascular Physiology
Smith, C. M., M.D.	University of Illinois College of Medicine Chicago, Illinois	Neuropharmacology
Stern, J. A., Ph.D.	Washington University School of Medicine St. Louis, Missouri	Physiological Psychology
Van der Helm, D., Ph.D.	University of Oklahoma Norman, Oklahoma	Chemistry, Biochemistry
Ward, A. A., M.D.	University of Washington, School of Medicine Seattle, Washington	Neurophysiology
Weiss, B., Ph.D.	Johns Hopkins University Baltimore, Maryland	Psychology, Pharmacology

Name and Degree	College or Institution and Location	Field of Interest
Wilson, W. A., M.D.	Bryn Mawr College Bryn Mawr, Pennsylvania	Physiological Psychology
Woodbury, J. W., Ph.D.	University of Washington Seattle, Washington	Myocardiac Physiology
Wootton, P.	Tumor Institute of the Swedish Hospital Seattle, Washington	Biophysics
Zablow, L., M.D.	College of Physicians and Surgeons Columbia University New York City, New York	Neurology

Appendix 6

LINC Participants

Dr. Ernst O. Attinger Dr. Antharvedi Anne'	The Presbyterian Hospital in Philadelphia Research Department
Dr. Donald S. Blough	Brown University Department of Psychology
Dr. C. Alan Boneau	Duke University Department of Psychology
Dr. Sidney Goldring Mr. Lloyd Simpson (Dr. James L. O'Leary)	Washington University School of Medicine Department of Psychiatry and Neurology
Dr. Fred S. Grodins Dr. James E. Randall	Northwestern University Medical School Department of Physiology
Dr. J. E. Hind Dr. C. Daniel Geisler	University of Wisconsin Laboratory of Neurophysiology
Mr. Lee Hundley (Dr. Joshua Lederberg)	Stanford University Medical School Department of Genetics
Dr. John C. Lilly Mr. Ben Locke	Communication Research Institute Miami, Florida
Dr. George S. Malindzak Dr. Fred Thurstone	Wake Forest College, Bowman Gray School of Medicine Department of Physiology
Dr. Gian F. Poggio Dr. Gerhard Werner (Dr. Vernon Mountcastle)	The Johns Hopkins University School of Medicine Department of Physiology
Dr. Bernard Weiss	The Johns Hopkins University School of Medicine Department of Pharmacology and Experimental Therapeutics
Dr. J. Walter Woodbury Dr. Albert Gordon	University of Washington Department of Physiology and Biophysics

ROSTER

First Assembly Period

LINC Evaluation Program

Donald S. Blough, Ph.D.	Brown University, Department of Psychology
C. Daniel Geisler, Sc.D.	The University of Wisconsin, Laboratory of Neurophysiology
Joseph E. Hind, Ph.D.	The University of Wisconsin, Laboratory of Neurophysiology
Lee Hundley, B.S.	Stanford University, Department of Genetics
John C. Lilly, M.D.	Communication Research Institute
Ben Locke	Communication Research Institute
Gian F. Poggio, M.D.	The Johns Hopkins University, Department of Physiology
Bernard Weiss, Ph.D.	The Johns Hopkins University, Department of Pharmacology and Experimental Therapeutics
Gerhard Werner, M.D.	The Johns Hopkins University, Department of Physiology

LINC Evaluation Program

Second Assembly Phase Participants

Fred S. Grodins, M.D. James E. Randall, Ph.D.	Northwestern University Department of Physiology
C. Alan Boneau, Ph.D.	Duke University Department of Psychology
Ernst O. Attinger, M.D. Antharvedi Anné, Ph.D.	Research Department Presbyterian Hospital in Philadelphia
George S. Malindzak, Ph.D. Fredrick L. Thurstone, Ph.D.	Wake Forest College, The Bowman Gray School of Medicine, Winston-Salem, North Carolina
Sidney Goldring, M.D. Lloyd N. Simpson	Washington University School of Medicine Department of Psychiatry and Neurology
J. Walter Woodbury, Ph.D. Albert M. Gordon, Ph.D.	University of Washington Department of Physiology and Biophysics

Appendix 7

INTERDEPARTMENTAL MASSACHUSETTS INSTITUTE OF TECHNOLOGY

CENTER DEVELOPMENT OFFICE
FOR COMPUTER TECHNOLOGY IN THE BIOMEDICAL SCIENCES
292 MAIN STREET, CAMBRIDGE 39, EXT. 5841

14 August 1963

To: Central File

From: T. T. Sandel

As of 13 August 1963 all LINCs from the first assembly phase including the Killam LINC had been delivered in apparently good order. The date of departure was nominally 26 July 1963 and the dates of arrival were as follows:

D. S. Blough, Brown University, Providence, Rhode Island -- 27 July 1963

G. F. Poggio, G. Werner } Johns Hopkins University, Baltimore, Maryland -- 2 August 1963

B. Weiss, Johns Hopkins University, Baltimore, Maryland -- 2 August 1963

J. Lilly, Communication Research Institute, Miami, Florida -- 9 August 1963

K. Killam, J. Hance } Stanford University, Palo Alto, California -- 9 August 1963

L. Hundley (J. Lederberg), Stanford University, Palo Alto, California -- 9 August 1963

J. Hind, C. Geisler } University of Wisconsin, Madison, Wisconsin -- 13 August 1963

No damage has been noted in any case which might be attributed to transportation via furniture van. The only malfunctions which have been noted fall into trivial categories and have been easily remedied.

INTERDEPARTMENTAL MASSACHUSETTS INSTITUTE OF TECHNOLOGY

CENTER DEVELOPMENT OFFICE
FOR COMPUTER TECHNOLOGY IN THE BIOMEDICAL SCIENCES
292 MAIN STREET, CAMBRIDGE 39, EXT. 5841

30 September 1963

To: Central File

From: T. T. Sandel

As of 30 September 1963 all LINCs from the second assembly phase including the Stacy LINC had been delivered in apparently good order. The date of departure was nominally 30 August 1963 and the dates of arrival were as follows:

E. O. Attinger, A. Anne' } Presbyterian Hospital in Philadelphia, Pennsylvania -- 6 September 1963

F. S. Grodins, J. E. Randall } Northwestern University, Chicago, Illinois -- 11 September 1963

C. A. Boneau Duke University, Durham, North Carolina -- 12 September 1963

R. W. Stacy, N. R. Bell } Institute of Statistics, Raleigh, North Carolina -- 12 September

G. S. Malindzak, F. L. Thurstone } Wake Forest College, The Bowman Gray School of Medicine Winston-Salem, North Carolina -- 12 September 1963

S. Goldring, L. N. Simpson } Washington University, St. Louis, Missouri -- 17 September 1963

J. W. Woodbury, A. M. Gordon } University of Washington, Seattle, Washington -- 30 September 1963

G. Malindzak reports that one of his console cases was cracked in transit. No other damage has been reported.

Appendix 8

An Introduction to Binary Numbers and Binary Arithmetic

Irving H. Thomae

28 June 1963

An Introduction to Binary Numbers and Binary Arithmetic

From a pragmatic viewpoint, any numerical notation or number system is merely a code for representing quantities - statements about "how many." In other words, a number system is a language in which topics like counting and arithmetic can be discussed conveniently. We may not expect that such a language will be unique. There may be, and in fact there is, a whole family of number systems, and the particular number system used by a particular digital computer is, in this sense, that computer's "language." While we do arithmetic in the decimal system, LINC and many other computers use the binary number system. Before explaining binary, let us recall what is meant by a "decimal" system.

Everyone learns in grade school that a decimal number such as 7,432 represents "two ones, three tens, four hundreds, and seven thousands." Reading from right to left, in other words, the successive columns are ascending powers of ten: $10^0(=1)$, 10^1, 10^2, 10^3, etc. The system is based on ten, as the name implies, and there are ten different symbols used, 0 through 9.

But there is nothing to prevent us from using some other number as a base, or radix. The addition tables etc. would have to be rewritten, since the same quantities would be differently encoded, but two plus two, by any name, must still be four, even though we may write "$\beta + \beta = \delta$", or "10 + 10 = 100." In this paper, we will use spelled-out names of numbers to refer to the quantities they represent, independent of particular number systems. Thus, "two" is an invariant. It always means the number

of dots in this circle: ⊙. The mark "2," however, is undefined in some number systems, including binary; and the mark "10" has a different meaning in each different number system.

The binary system is based on the radix two. This means that there need be only two symbols, conventionally taken as 0 and 1. This is why computers use it, since an on-off, or two-state, device is much simpler than a ten-state device.

Reading from the right end of a binary number, successive columns are ones, twos, fours, eights, etc., - $2^0(=1)$, 2^1,2^2,2^3, etc. - ascending powers of two. Thus, the number 11001 represents, reading from right, "one one plus no twos plus no fours plus one eight plus one sixteen," or twenty-five. It must be admitted that binary numbers are less compact than decimal, but for computer use, we will see that this disadvantage is far outweighed by the advantages.

Compare the following numbers:

DECIMAL			BINARY						
10^2	10^1	10^0	2^6	2^5	2^4	2^3	2^2	2^1	2^0
hundreds	tens	ones	sixty-fours	thirty-twos	sixteens	eights	fours	twos	ones
0	0	1	0	0	0	0	0	0	1
0	0	2	0	0	0	0	0	1	0
0	0	3	0	0	0	0	0	1	1
0	0	4	0	0	0	0	1	0	0
0	0	5	0	0	0	0	1	0	1
0	0	6	0	0	0	0	1	1	0
0	0	7	0	0	0	0	1	1	1
0	0	8	0	0	0	1	0	0	0
0	0	9	0	0	0	1	0	0	1
0	1	0	0	0	0	1	0	1	0
0	1	1	0	0	0	1	0	1	1
0	1	2	0	0	0	1	1	0	0
0	1	3	0	0	0	1	1	0	1
0	1	4	0	0	0	1	1	1	0
0	1	5	0	0	0	1	1	1	1
0	2	0	0	0	1	0	1	0	0
0	2	6	0	0	1	1	0	1	0
0	6	3	0	1	1	1	1	1	1
0	6	4	1	0	0	0	0	0	0
1	1	7	1	1	1	0	1	0	1

Fractions are represented in the same way. Columns to the right of a decimal point represent increasingly negative powers of ten (tenths, hundredths, thousandths, or 10^{-1}, 10^{-2}, 10^{-3}, etc.). Similarly, to the right of the binary point we have halves, quarters, eighths - 2^{-1}, 2^{-2}, 2^{-3}, etc. Any fraction can be represented in this form. For instance, .1011 is "1 half plus no fourths plus 1 eighth plus 1 sixteenth," or eleven sixteenths, .6875. Similarly, 101.011 is 5 3/8, or 5.375.

The tables of powers of two attached, especially the positive powers, should be at one's mental finger tips.

We will often refer to the columns of a binary number as "bits." Strictly speaking, a "bit" is any item of yes-or-no information, but in practice this distinction will usually be unimportant. We also frequently name a bit in a number by the power of two represented. Thus, the "0-bit" is the right-most bit, representing 2^0 or 1; "bit 4" is the fifth from the right, representing 2^4 or sixteen.

The numbers listed on the preceding page illustrate two important points about number systems. Consider first the counting process with respect to one column of a decimal number. As 1's are added, the column "fills up" until 9 is reached. This is the maximum capacity, so when the next 1 is added we must return our column to 0 and carry 1 to the next higher column.

In the binary system, however, a given column's value may only be 0 or 1, so every second time a bit receives a 1 it must clear and carry to the next. For a given bit, then, counting is a process of alternating, or "flipping," between "0" and "1", originating (sending out) a carry every time it reverts from "1" to "0."

In either system, when all columns are filled to capacity, the next "1" added will require a new column. In decimal, we see this happen in going from 9 to 10, or 99 to 100; in binary this happens, for example, between seven and eight, fifteen and sixteen, or sixty-three and sixty-four.

Notice also that it is always extremely easy to multiply by a power of the radix. In decimal, we may multiply by ten by shifting the entire number left one place, or by 10^n by shifting left n places. Correspondingly, in binary we can multiply by two by shifting left one place, or by 2^n by shifting left n places. Compare three, six, and twelve in binary with three, thirty, and three hundred in decimal; or thirteen and twenty-six in binary with thirteen and one hundred-thirty in decimal:

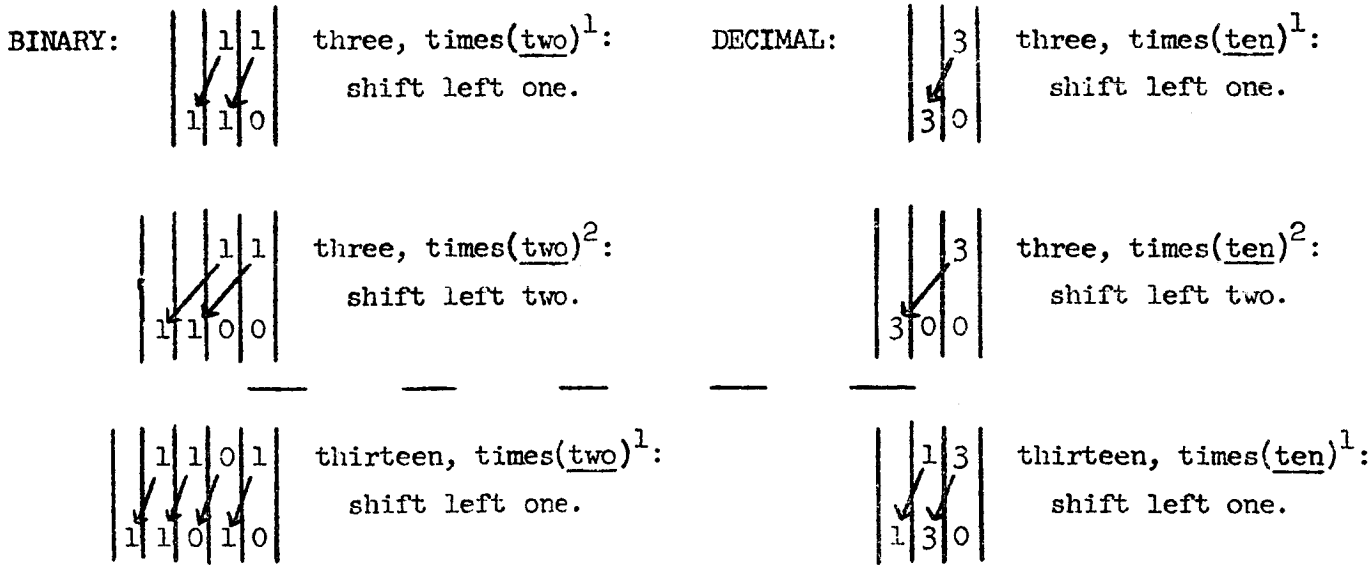
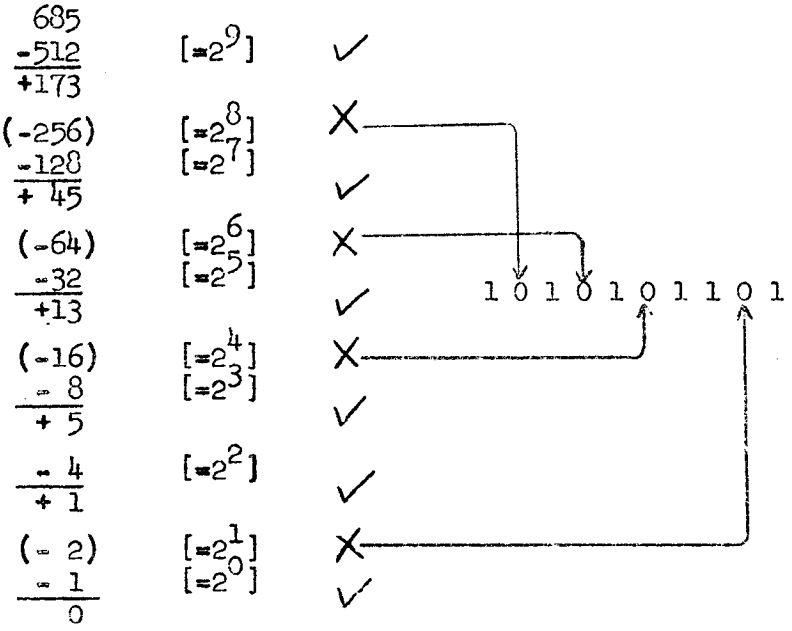


FIGURE 1. Multiplication by the radix as a shifting process.

The process of "translating," or reconverting from binary to decimal is obvious; it might be helpful to describe decimal-to-binary conversion

explicitly. Starting with the largest possible power of two, we attempt to subtract successively smaller powers of two from the current remainder and get a positive result. For each successful subtraction a 1 is recorded, otherwise, a 0. Thus the decimal number 685 converts as follows:



ADDITION

Binary addition is very simple. The basic table has only four entries, compared to one hundred in decimal:

Decimal:

+	0	1	2	3 ...
0	0	1	2	3
1	1	2	3	4
2	2	3	4	5
	etc.			

Binary:

+	0	1
0	0	1
1	1	10

Notice that for a particular bit, 1+1 gives 0 with a carry. This we have just seen in considering the counting process. Indeed, from the viewpoint of a particular bit, addition is always basically a counting process.

We may illustrate with an example.

	1 1 0 1 0
	1 0 1 1 0
column sums	0 1 1 0 0
primary carries	1 1
first carry sum	1 0 1 0 0 0
secondary carries	1
secondary carry sum	1 0 0 0 0 0
tertiary carries	1
result	1 1 0 0 0 0

Of course, in doing the sum one would normally add in the carries as they appeared, but this form shows what is going on more clearly.

MULTIPLICATION

Binary multiplication is, if anything, even simpler than addition. The basic table is:

	0	1
0	0	0
1	0	1

1 times 1 is 1, and anything else is 0. It is easy enough to combine this with the standard methods. Compare decimal and binary multiplication:

a.

$$\begin{array}{r} 356 \\ \underline{708} \\ 2848 \\ 0000 \\ \underline{2492} \\ 252048 \end{array}$$

b.

$$\begin{array}{r} A \\ \times\ B \\ \hline C \end{array}$$

Decimal:

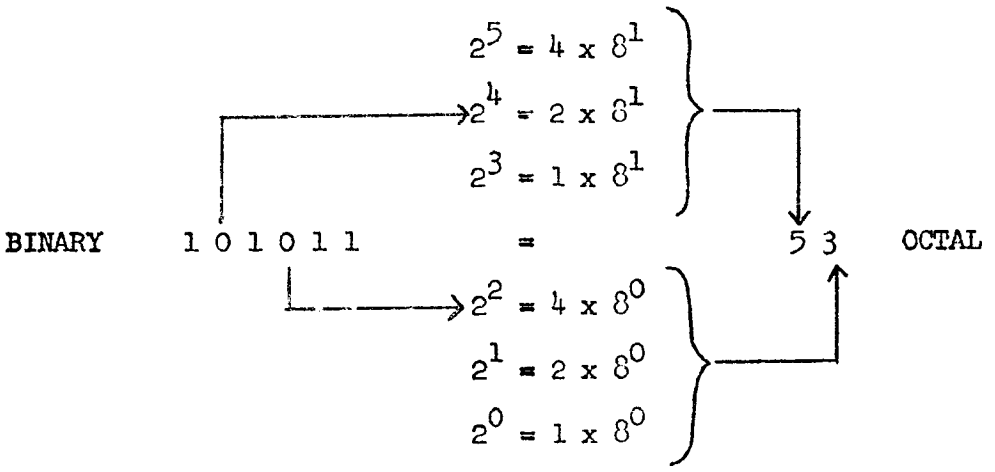
$$\begin{array}{r} 29 \\ \underline{21} \\ 29 \\ \underline{58} \\ 609 \end{array}$$

Binary:

$$\begin{array}{r} 11101 \\ \underline{10101} \\ 11101 \\ 00000 \\ 11101 \\ 00000 \\ \underline{11101} \\ 1001100001 \end{array}$$

Of course we normally omit the rows of zeros. But notice that in binary, multiplication by each multiplier digit is reduced to the decision whether or not to copy the shifted multiplicand. So, in this example, we take $(1) \cdot [(2^0)(A)] + (0) \cdot [(2^1)(A)] + (1) \cdot [(2^2)(A)] + (0) \cdot [(2^3)(A)] + (1) \cdot [(2^4)(A)]$, or in all, twenty-one times A, or B times A, which is exactly what we want.

By now it should be quite clear that binary arithmetic is both simple and cumbersome. There is available a very convenient way to avoid the awkward chains of ones and zeros. We introduce <u>another</u> number system, <u>octal</u>, which is based on <u>eight</u>. This system has 8 characters, for which we use the Arabic numerals 0 through 7. Interconversion between binary and octal can be done by sight, since a group of three binary digits is completely equivalent to <u>one</u> octal digit. This of course is so because $8 = 2^3$. So we have this equivalence:



BINARY	OCTAL
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

See also the tables of powers of two.

A binary number grouped in sets of 3 bits each can thus be read off in octal, a far more convenient notation.

Compare these numbers:

DECIMAL	BINARY	OCTAL
5	101	5
8	1 000	10
9	1 001	11
12	1 100	14
15	1 111	17
16	10 000	20
29	11 101	35
32	100 000	40
40	101 000	50
54	110 110	66
100	1 100 100	144
3739	111 010 011 011	7233

One can clearly also do arithmetic in octal, and although LINC actually operates in binary, it is customary and proper to use octal almost exclusively when programming and operating the computer.

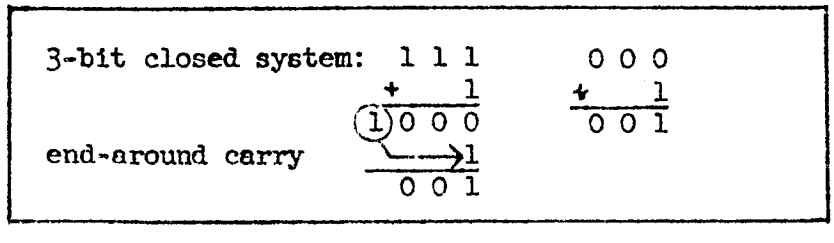
The addition and multiplication tables are perfectly straightforward, except that the digits 8 and 9 are missing. For convenience they have been appended. A brief glance at them will indicate that the same counting processes hold as in any other system: when the capacity of a particular column is exceeded, clear it and carry.

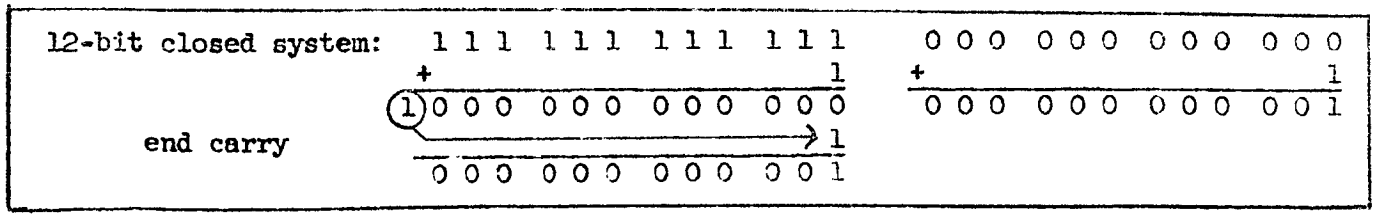
One pitfall to be avoided carefully is that octal numbers look, superficially at least, much like ordinary numbers. The complete absence of 8's and 9's may not be immediately evident, and much confusion can result. Therefore, wherever ambiguity seems possible, numbers will be written with the subscript "8" or "10" to indicate "octal" or "decimal."

COMPLEMENT ARITHMETIC

Up to this point we have assumed that we could represent any number, no matter how large. In a digital computer, each digit is represented by some kind of physical hardware, such as a wire, a "flip-flop," or a light-bulb. The computer thus necessarily has a finite range of numbers, determined by the number of bits available. For instance, LINC has twelve bits, so the largest possible number which could normally be represented is 4095_{10} (7777_8). If we add 1 to this number, we ought to get 1 000 000 000 000_2 (10000_8), but only the rightmost twelve bits are represented, so the next number in sequence is 0. More simply, but in exactly the same way, if we had three bits available the biggest number possible would be 111 (7_8), followed again by 0. In that closed system 1, 9, and 17 are completely equivalent. They are said to be equal "modulo 8;" in other words, they all give the same remainder when divided by 8.

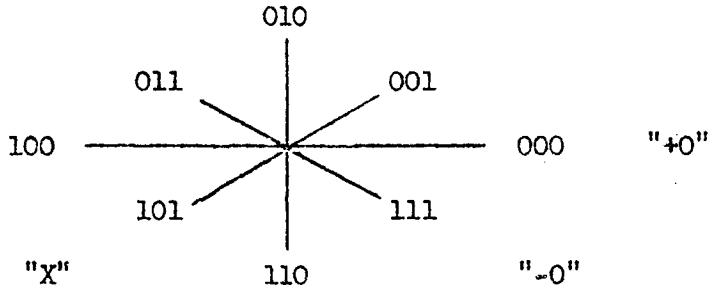
Suppose now we arrange things so that, in our simple closed system, the extra carry generated when we add 1 to 7 is <u>brought around the end</u> - as an "<u>end-carry</u>" - and added in at the 0-bit. When this occurs, the result is 001 - as if we had added 1 to 0 instead of to 7!





In fact, in end-carry n-digit binary arithmetic, the number composed of n 1's behaves exactly like 0. It is customarily referred to as "minus zero" to distinguish it from the more familiar form which is then referred to as "plus zero."

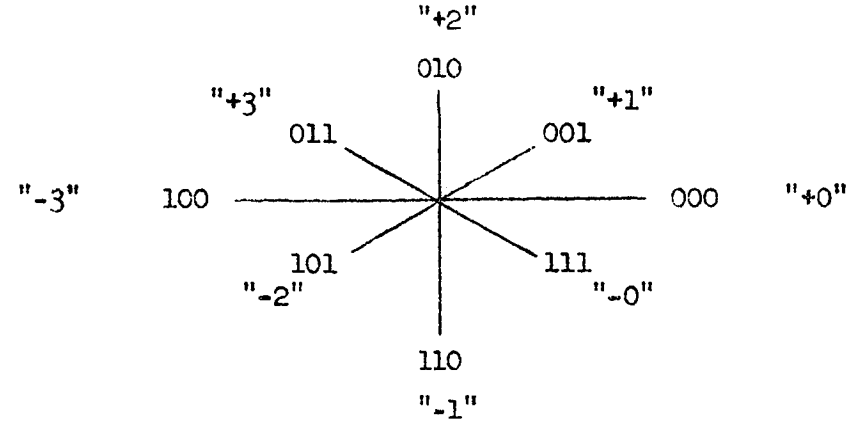
Continuing in our 3-digit end-carry system, we consider the following arrangement of numbers:



These are, of course, all the possible numbers we may have.

Define positive rotation counter clockwise (↻). Counting around the wheel, the position "X" is then plus five. However, if we start at "-0" and count clockwise, it becomes minus two. Try adding this "-2" to +3, using end-around carry. (Count around the wheel treating +0 and -0 as one point.) The result is +1.

It will be found that the following designations can be assigned:



These definitions permit subtraction, if we limit ourselves to a system of just the numbers 0,1,2,3. In effect we have made the leftmost bit represent the sign of the number. If it is a "1", the number is presumed to be negative, and is counted down from minus zero (111), instead of up from plus zero (000).

Notice in the diagram that -2(101) has 1's where +2(010) has 0's, and vice versa. The process of replacing 1's with 0's and 0's with 1's is called complementing, and in a 3-bit system, 101 is the complement of 010. So, to encode a negative number, we complement the corresponding positive binary number.

Returning now to LINC's 12-bit numbers, we restrict ourselves to numerical use of 11 bits. To code a negative number, we complement it. Here, too, complementing turns out to be equivalent to counting down from -0. Since the binary magnitude of numbers will have a zero in the leftmost bit, complementing renders bit 11 a one. This bit is therefore a sign indicator, and LINC can "find out" whether a number is positive or negative simply by testing it. If the 11-bit is 0, the number is positive; if 1, it is negative. Furthermore, if we never try to give numerical significance to the sign bit, we can subtract numbers by adding their complements, using end-around carry.

EXAMPLE:

Decimal	Binary	Octal
1978	011 110 111 010	3672
-1568	-011 000 100 010	-3042
410	011 110 111 010	3672
Add complement of subtrahend:	100 111 011 101	4735
End-around carry:	(1) 000 110 010 111	(1) 0627
	1	1
	000 110 011 000	0630

Notice that in octal, we may form a complement by subtracting each digit of the number to be complemented from 7. Then, by using end-around carry, we get the same result as we did in binary.

Counting now is rather odd if one exceeds the allowed eleven numerical bits. For, the next number after 011 111 111 111, (3777_8), the largest positive number, is 100 000 000 000, the biggest (in magnitude) negative number ($4000_8 = -3777_8$).

DIVISION

The last and perhaps most confusing operation is long division. Division is the process of finding out how many times the divisor is contained in the dividend. At bottom, therefore, it is an elaborate method of subtracting and counting, although the familiar procedures tend to obscure this.

For example, in the simple division $2\overline{)9}$, we all know the quotient is 4 and the remainder is 1. But if we didn't know that, we could find by subtracting 2 repeatedly, counting the number of times we were successful. When, after such a series, the result turns up negative, we know we have subtracted once too often. The correct remainder is then recovered by adding back the divisor once, and the correct quotient is one less than the total number of subtractions we have executed.

Let us illustrate this with another very simple example, $3\overline{)14}$:

Operations	Count	
14		
$\underline{-\ 3}$ 11	1	
$\underline{-\ 3}$ 8	2	
$\underline{-\ 3}$ 5	3	
$\underline{-\ 3}$ 2	4	
$\underline{-\ 3}$ -1	5	Negative result, so add back the divisor.
$\underline{+\ 3}$ 2	5-1 = 4	

Quotient 4, remainder 2

This obviously is impractical for large quotients, and so the familiar long division uses a very important shortcut.

Consider this example:

$$
\begin{array}{r}
142 \\
28\overline{)3979} \\
28 \\
\hline
117 \\
112 \\
\hline
59 \\
56 \\
\hline
3
\end{array}
$$

In the first step, we actually divide not by 28, but by 2800. To obtain the right answer for this problem, that result is automatically multiplied by 100 when it is put in the third column from right in the answer. That is, the "1" in the quotient represents $\left(\frac{3979}{28 \times 100} \right) \times 100$.

The remainder obtained is really 1179. In the second set of steps 1179 is divided by 28×10^1, and the result, 4, is multiplied by 10^1 when it is put in the second quotient column. Finally, 59 is divided by 28×10^0, and the result, 2, is multiplied by 1 and placed in the rightmost column to give the answer 142.

The same "shortcut" of taking out the divisor "a hundred at a time" can be used in the subtraction method too, as follows:

Operation	Comments	Count	Quotient
28/3979			
-3979	Divide by 28 x 100		
-2800			
1179		1	
+2800	Negative, so add back	2	
-1621	the divisor		
+2800		2-1 = 1	1 x 100
1179			
- 280	Now use 28 x 10 as	1	
899	divisor		
- 280		2	
619	Positive remainder,		
- 280	so keep going.		
339		3	
- 280		4	
59			
- 280			
- 221	Negative, so add back	5	
+ 280	the divisor again.		
59	This count is the 10's		
	digit of the quotient.	5-1 = 4	4 x 10
- 28	Now use 28 x 1 as divisor		
31		1	
- 28		2	
3	Positive, so keep going		
- 28			
- 25	Negative - back up		
+ 28			
+ 3		3-1 = 2	2 x 1

The final quotient is 100 + 40 + 2 = 142, and the final remainder is 3.

Of course, the process of multiplying the divisor by various powers of ten is customarily accomplished purely by shifting it along beneath the dividend, and the zeros have been filled in here purely for clarity.

Notice that, if we wished, we could shift the dividend left instead of shifting the divisor right. Their positions relative to each other will be unchanged and if we don't get our quotient score-keeping mixed up,

the result will be exactly the same. This is convenient in a finite number system like LINC's, where shifting a number may mean discarding digits. It is then clearly better to discard higher-order current remainder bits, which are 0 anyway when they fall due for shifting "off into space."

We will not attempt to do binary division with complemented numbers. If either the divisor or the dividend is negative, we will re-complement it before dividing, and remember the sign.

As an illustration, let us find the quotient of two 6-bit binary fractions. As usual, the left-most bit is the sign; and we assume also that the binary point lies directly to its right. With the binary points of both divisor and dividend in the same place, this is completely equivalent to dividing a pair of integers. However, use of the left-most bit as sign-bit requires that the divisor be greater than the dividend. For, if the quotient came out equal to or greater than 1, it would then be interpreted as a negative number, and this clearly would be wrong, since as we have already said, both the divisor and the dividend will always be positive.

Example: $\frac{1.10001}{0.10100} = ?$

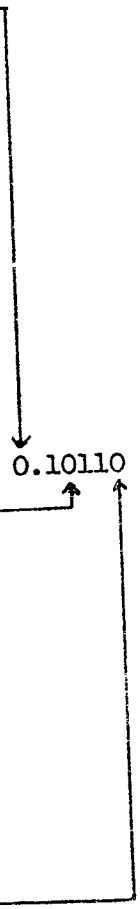
First, we note that the numerator is negative, so we must complement it, and remember to complement the quotient we get when we are all done.

So we have $0.10100/\overline{0.01110}$

First subtraction: (by adding complement of divisor)	001110 101011 111001	Negative: Record 0 in quotient, add back divisor. (This was expected, since the first quotient digit is a sign bit.)
add back:	+010100 001110	
Shift remainder left one:	011100	
Subtract:	101011 001000	Positive - record 1 in quotient, continue.
Shift remainder left one:	010000	
Subtract:	101011 111011	Negative - add back divisor, record 0 in quotient.
Shift:	100000	
Subtract:	101011 001100	Positive - record 1 in quotient, continue.
Shift:	011000	
Subtract:	101011 000100	Positive - record 1, continue.
Shift:	001000	
Subtract:	101011 110011	Negative - record 0, add back.
	010100 001000	

etc.

Quotient: 0.10110



The first 6 bits of the quotient are therefore 0.10110.

Complementing, the final result is $\frac{1.10001}{0.10100} = 1.01001$

The reader may verify that in decimal this would read $\frac{-.4375}{.6250} = -.700$,

and that the binary equivalent of .700 is 0.10110011.....

Now, there is one possible "shortcut" peculiar to binary. We have seen that when subtracting of the divisor gives a negative result, the divisor must be added back before shifting and subtracting again. In binary, upon getting a negative result, we can shift <u>first</u> and <u>then</u> add the divisor.[1] However, when we shift before restoring, we are working with a complement, and cannot discard bits shifted "off the left end." In order to make the end-around carry come out right, it is necessary to bring the shifted bits around to the right and fill them in there. In LINC, this is called <u>rotation</u> to distinguish it from ordinary <u>shifting</u>.

1 We are shifting the remainder left, which multiplies it by two. So, if R is the number from which we just subtracted, and D is the divisor, the negative result is (R-D). It is obvious that

$2[(R-D) + D] - D = 2(R-D) + D.$

Here is the same example, using the "shortcut":

0.10100/0.01110

Subtract:	001110	
	101011	
	111001	Negative: Record 0 in quotient, rotate left one place.
rotate:	110011	
add:	010100	
	001000	Positive: record 1 in quotient, continue.
rotate:	010000	(Notive that rotating a positive number
subtract:	101011	left one place is indistinguishable
	111011	from shifting it left one place)
		Negative: so record 0, rotate, add.
rotate:	110111	
add:	010100	
	001100	Positive: record 1, continue.
rotate:	011000	
subtract:	101011	
	000100	Positive: record 1, continue.
rotate:	001000	
subtract:	101011	
	110011	Negative: record 0, continue.
rotate:	100111	
add:	010100	
	111011	
	etc.	Quotient: 0.10110

POWERS OF TWO AND EIGHT

POSITIVE POWERS			DECIMAL EQUIVALENTS
2^0	8^0		1
2^1		2×8^0	2
2^2		4×8^0	4
2^3	8^1		8
2^4		2×8^1	16
2^5		4×8^1	32
2^6	8^2		64
2^7		2×8^2	128
2^8		4×8^2	256
2^9	8^3		512
2^{10}		2×8^3	1,024
2^{11}		4×8^3	2,048
2^{12}	8^4		4,096
2^{13}		2×8^4	8,192
2^{14}		4×8^4	16,384
2^{15}	8^5		32,768

POWERS OF TWO AND EIGHT

NEGATIVE POWERS			DECIMAL EQUIVALENTS
2^{0}	8^{0}		1.0
2^{-1}		4×8^{-1}	.5
2^{-2}		2×8^{-1}	.25
2^{-3}	8^{-1}		.125
2^{-4}		4×8^{-2}	.0625
2^{-5}		2×8^{-2}	.03125
2^{-6}	8^{-2}		.015625
2^{-7}		4×8^{-3}	.0078125
2^{-8}		2×8^{-3}	.00390625
2^{-9}	8^{-3}		.001953125
2^{-10}		4×8^{-4}	.0009765625
2^{-11}		2×8^{-4}	.00048828125
2^{-12}	8^{-4}		.000244140625
2^{-13}		4×8^{-5}	.0001220703125
2^{-14}		2×8^{-5}	.00006103515625
2^{-15}	8^{-5}		.000030517578125

OCTAL ADDITION

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

OCTAL MULTIPLICATION

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

LINC LIT ORDER CODE

Mary Allen Wilkes

31 July 1963

HLT	0000		HLT
HALT. The computer halts, the Run light on the control console goes off, and the ground rings.			

CLR	0011	8 μsec	CLR
CLEAR. Clear the Accumulator and the link bit. 0 → C(ACC); 0 → C(L).			

MSC 13	0013	8 μsec	MSC 13
WRITE GATE ON. The write gate for marking tapes is turned on if, and only if, the Mark push button on the console has been depressed. The instruction is only used when generating LINC tapes.			

ATR	0014	8 μsec	ATR
ACCUMULATOR TO RELAY. The contents of the right half (bits 0-5) of the Accumulator replace the contents of the relay register. The Accumulator is left unchanged.			

RTA	0015	8 μsec	RTA
RELAY TO ACCUMULATOR. The contents of the relay register replace the contents of the right half (bits 0-5) of the Accumulator. The left half of the Accumulator is cleared. The relay register is left unchanged.			

NOP	0016	8 μsec	NOP
NO OPERATION. This instruction provides a delay of 8 μsecs. before proceeding to the next instruction. It does nothing.			

CCM	0017	8 μsec	CCM
COMPLEMENT. Complement the number in the Accumulator. C(ACC) → C(ACC).			

II. SHIFT CLASS INSTRUCTIONS

ROL i n	240 + 20i + n	16 μsec +	ROL

ROTATE LEFT. If i = 0, shift contents of the Accumulator n places to the left (n = 0, 1, ..., 17 octal), with bit 11 feeding bit 0. If i = 1, shift the Link - Accumulator combination n places to the left, with bit 11 feeding the Link bit and the Link bit feeding bit 0.

Time of execution: 16 μsec for n = 0, 1, 2, 3; 8 μsec additional for each additional 4 places or fraction thereof.

ROR i n	300 + 20i + n	16 μsec +	ROR

ROTATE RIGHT. If i = 0, shift contents of the Accumulator n places to the right (n = 0, 1, ..., 17 octal), with bit 0 feeding bit 11. If i = 1, shift the Link - Accumulator combination n places to the right, with bit 0 feeding the Link bit and the Link bit feeding bit 11.

Time of execution: 16 μsec for n = 0, 1, 2, 3; 8 μsec additional for each additional 4 places or fraction thereof.

SCR i n	340 + 20i + n	16 μsec +	SCR

SCALE RIGHT. If i = 0, shift the contents of the Accumulator n places to the right (n = 0, 1, ..., 17 octal), with bit 11 (the sign bit) unchanged and bits shifted out of bit 0 lost. If i = 1, shift the Accumulator as above; the last bit shifted out of bit 0 will be saved in the Link bit.

Time of execution: 16 μsec for n = 0, 1, 2, 3; 8 μsec additional for each additional 4 places or fraction thereof.

III. FULL-ADDRESS CLASS INSTRUCTIONS

ADD X	2000 + X	16 μsec	ADD
ADD. Add the contents of memory register X ($0 \leq X \leq 1777$) to the contents of the Accumulator, leaving the result in the Accumulator, C(X) + C(ACC) → C(ACC), using 12-bit binary addition with end-around-carry. Register X is unchanged.			

STC X	4000 + X	16 μsec	STC
STORE-CLEAR. Copy the contents of the Accumulator into memory register X($0 \leq X \leq 1777$) and then clear the Accumulator. C(ACC) → C(X), 0 → C(ACC).			

JMP X	6000 + X	16 μsec*	JMP
JUMP. Take the next instruction from memory register X ($0 \leq X \leq 1777$) and continue to execute instructions in sequence starting with register X. The address X replaces the contents of the Program Counter and, unless X = 0, the original contents of the Program Counter increased by 1 and prefixed by the code for JMP replace the contents of memory register 0. If C(PC) = p, then X → C(PC) and JMP p+1 → C(0) unless X = 0. If X = 0, then 0 → C(PC).			
*Execution time: For X = 0, 8 μsec; for X ≠ 0, 16 μsec.			

IV. INDEX CLASS INSTRUCTIONS

TABLE I. ADDRESSING IN INDEX CLASS INSTRUCTIONS

i	β	Y	t μsec	Comment
0	0	X(p + 1)	16	
1	0	p + 1	8	
0	1≤β≤17	X(β)	16	
1	1≤β≤17	X(β) + 1	16	X(β) + 1 → X(β)

The time t μsec must be added to the execution time to get the total instruction time. Y is the address of the register which holds the operand used by the instruction. The instruction is assumed to be in register p.

LDA i β	1000 + 20i + β	(8 + t) μsec*	LDA

LOAD ACCUMULATOR. Copy the contents of memory register Y (*see Table I) into the Accumulator. C(Y) → C(ACC). Register Y is unchanged.

STA i β	1040 + 20i + β	(8 + t) μsec*	STA

STORE ACCUMULATOR. Copy the contents of the Accumulator into memory register Y (*see Table I). C(ACC) → C(Y). The Accumulator is unchanged.

ADA i β	1100 + 20i + β	(8 + t) μsec*	ADA

ADD TO ACCUMULATOR. Add the contents of memory register Y (*see Table I) to the contents of the Accumulator, leaving the result in the Accumulator. C(Y) + C(ACC) → C(ACC), using 12-bit binary addition with end-around-carry. Register Y is unchanged.

ADM i β	1140 + 20i + β	(16 + t) μsec*	ADM

ADD TO MEMORY. Add the contents of memory register Y (*see Table I) to the contents of the Accumulator, leaving the result in the Accumulator and in register Y. C(Y) + C(ACC) → C(ACC) and → C(Y), using 12-bit binary addition with end-around-carry.

LAM i β	1200 + 20i + β	(16 + t) μsec*	LAM

LINK-ADD TO MEMORY. First add the contents of the Link bit (the integer 0 or 1) to the contents of the Accumulator leaving the sum in the Accumulator, using 12-bit binary addition with the end-carry, if any, replacing the contents of the Link bit; (if no end-carry arises, clear the Link bit). Next add the contents of register Y (*see Table I) to the contents of the Accumulator with the end-carry, if any, replacing the contents of the Link bit (if no end-carry arises, the Link bit is unchanged), leaving the 12-bit result in the Accumulator and in register Y.

MUL i β	1240 + 20i + β	(104 + t) μsec*	MUL

MULTIPLY. Multiply the contents of register Y (*see Table I) by the contents of the Accumulator, and leave the result in the Accumulator. The values in Y and in the Accumulator are treated as signed, 11-bit, ones' complement numbers, and are multiplied together to form a 22-bit product. They may be interpreted as either fractions or integers: if bit 11 (the h-bit) of the address Y is a one, they are treated as fractions whose binary points are between bit 11 (the sign bit) and bit 10. In this case the most significant 11 bits of the 22-bit product are left with the proper sign in the Accumulator. If bit 11 of the address Y is a zero, the values are treated as integers, and the least significant 11 bits of the 22-bit product are left with the proper sign in the Accumulator. When i = 1 and β = 0, bit 11 of the address Y is assumed to be zero, and the values are treated as integers. Register Y is unchanged. **The sign of the product is left in the Link bit.**

SAE i β	1440 + 20i + β	(8 + t) μsec*	SAE

SKIP IF ACCUMULATOR EQUALS. If the contents of the Accumulator exactly match the contents of memory register Y (*see Table I) then skip the next instruction (actually, skip the first register of the next instruction). Otherwise, go on to the next instruction in sequence. C(ACC) and C(Y) are unchanged in either case.

SRO i β	1500 + 20i + β	(8 + t) μsec*	SRO

SKIP AND ROTATE. If the rightmost bit of the contents of memory register Y (*see Table I) is a zero, then skip the next instruction (actually, skip the first register of the next instruction). Otherwise, go on to the next instruction in sequence. In either case, rotate the contents of register Y one place to the right and replace in register Y. The Accumulator is unaffected.

BCL i β	1540 + 20i + β	(8 + t) μsec*	BCL

BIT CLEAR. For each bit of the contents of register Y (*see Table I) which is a one, clear the corresponding bit of the contents of the Accumulator. Register Y and other bits in the Accumulator are unaffected.

BSE i β	1600 + 20i + β	(8 + t) μsec*	BSE

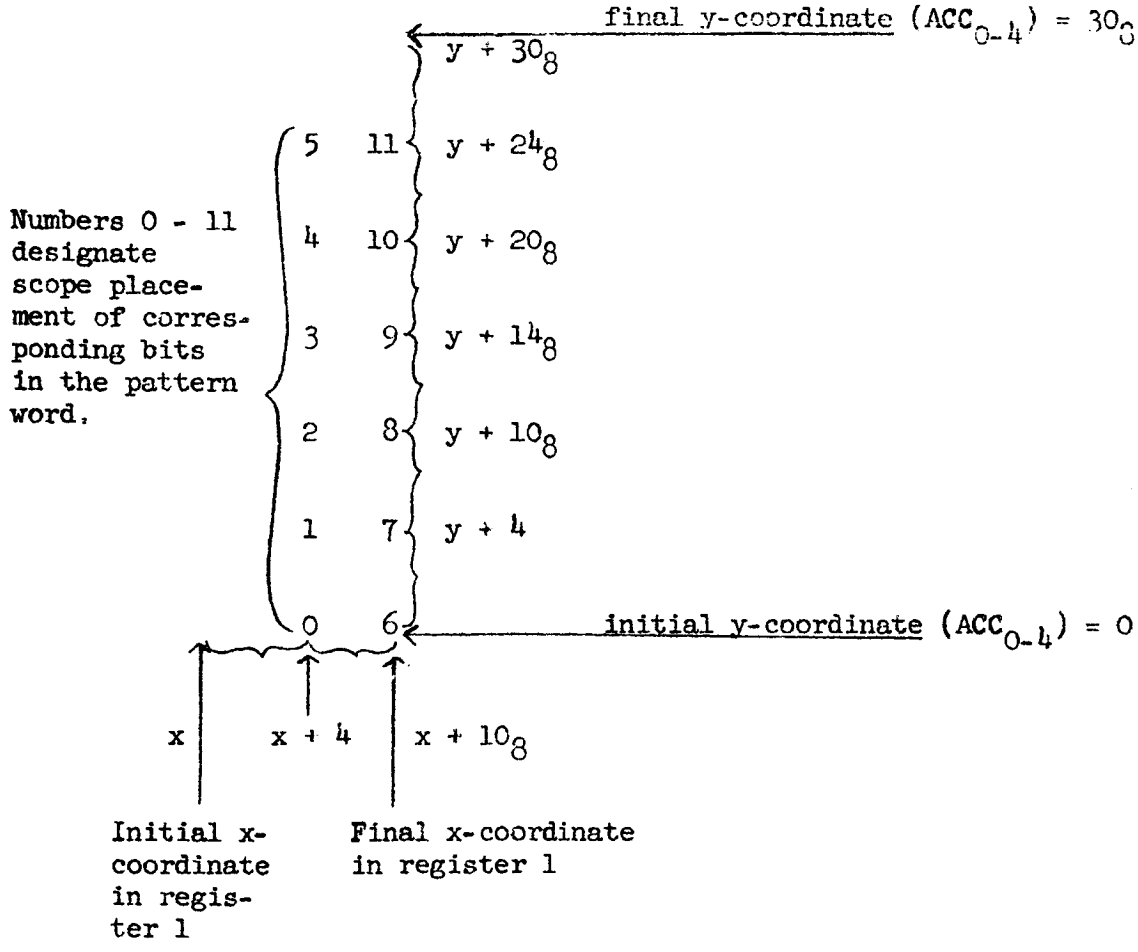
BIT SET. For each bit of the contents of register Y (*see Table I) which is a one, set to one the corresponding bit of the contents of the Accumulator. Register Y and other bits in the Accumulator are unaffected.

BCO i β	1640 + 20i + β	(8 + t) μsec*	BCO

BIT COMPLEMENT. For each bit of the contents of register Y (*see Table I) which is a one, complement the corresponding bit of the contents of the Accumulator. Register Y and other bits in the Accumulator are unaffected.

DSC 1 β	174) + 201 + β	(112 + t) μsec*	DSC

DISPLAY CHARACTER. Display, in a 2 x 6 grid, the pattern contained in register Y (*see Table I). The contents of register Y are examined from right to left beginning with bit zero, and for each bit found to be a one a point is displayed. The initial **x**-coordinate will be the contents of register 1, plus 4; the display channel is selected by the leftmost bit of register 1. The initial **y**-coordinate will be the contents of the Accumulator with the rightmost 5 bits (bits 0-4) set to zero by the computer. The initial coordinates specify the lower left position of the display; the computer proceeds from lower left to upper right. For each bit of register Y which is examined, +4 is added to the **y**-coordinate in the Accumulator. When the right 6 bits of register Y have been examined, the right 5 bits of the Accumulator are reset to zero and +4 is added to the **x**-coordinate in register 1. The procedure is then repeated for the left 6 bits of register Y. At the conclusion of the instruction the contents of register 1 have been incremented by 10 (octal), and the right 5 bits of the Accumulator are left equal to 30 (octal). Register Y is unchanged.



V. HALF-WORD CLASS INSTRUCTIONS

Table II ADDRESSING IN HALF-WORD CLASS INSTRUCTIONS

i	β	Y	h	t μsec	Comment
0	0	X(p + 1)	h(p + 1)	16	If h=0, operand is LH(X) If h=1, operand is RH(X)
1	0	p + 1	0	8	Operand is always LH(p + 1)
0	$1 \leq \beta \leq 17$	X(β)	h(β)	16	If h=0, operand is LH(X) If h=1, operand is RH(X)
1	$1 \leq \beta \leq 17$	X(β) + h(β)	$\bar{h}$(β)	16	If $\bar{h}$=0, operand is LH(X+1) If $\bar{h}$=1, operand is RH(X) h, j, X + h → C(β)

The time t μsec must be added to the execution time to get the total instruction time. Y is the address of the register holding the operand in the half designated by h. (h = 1: right half; h = 0: left half). The instruction is assumed to be in register p.

LDH i β	1300 + 20i + β	(8 + t) μsec*	LDH
LOAD HALF. Copy the contents of the designated half of register Y (*see Table II) into the right half of the Accumulator, clearing the left half of the Accumulator. Register Y is unchanged.			

STH i β	1340 + 20i + β	(8 + t) μsec*	STH
STORE HALF. Copy the contents of the right half of the Accumulator into the designated half of register Y (*see Table II). The Accumulator and the unused half of register Y are unchanged.			

SHD i β	1400 + 20i + β	(8 + t) μsec*	SHD
SKIP IF HALF DIFFERS. If the contents of the right half of the Accumulator differ from the contents of the designated half of register Y (*see Table II) then skip the next instruction (actually, skip the first register of the next instruction). Otherwise, go on to the next instruction in sequence. C(ACC) and C(Y) are unchanged in either case.			

VI. MISCELLANEOUS INSTRUCTIONS

TABLE III. ADDRESSING IN SET INSTRUCTION

i	n	Y	t μsec
0	$0 \leq n \leq 17$	X(p + 1)	0
1	$0 \leq n \leq 17$	p + 1	0

SET i n	40 + 20i + n	(24 + t) μsec*	SET

SET. Set register n equal to the contents of register Y (*see Table III). The Accumulator and register Y are unchanged. Take the next instruction from p + 2.

SAM i n	100 + 20i + n	8 μsec 24 μsec	SAM

SAMPLE. Sample the signal on one of 16 input channels selected by n. Put its binary conversion, ±177, into the Accumulator, extending the sign through bit 11. $0 \leq n \leq 7$ selects one of the potentiometers; $10 \leq n \leq 17$ selects one of the high speed inputs. If i = 0, the instruction requires 24 μsec. If i = 1, the computer goes on to the next instruction after 8 μsec; the conversion process in the Accumulator continues, however, for an additional 14 μsec. Therefore, care should be taken when instructions which affect the Accumulator follow a SAM with i = 1.

DIS i n	140 + 20i + n	32 μsec	DIS

DISPLAY. When i = 1, index the address part of register n (n = 0,1,..., 17 octal) by 1. Display one point whose x-coordinate is specified by the rightmost 9 bits of register n, and whose y-coordinate, $+377 \geq y \geq -377$ (octal) is specified by the contents of the Accumulator. Display via the channel selected by the leftmost bit of register n. The Accumulator remains unchanged.

XSK i n	200 + 20i + n	16 μsec	XSK

INDEX AND SKIP. If i = 1, increment the address part of register n by 1. If i = 0, do not index register n. In either case, skip the next instruction (actually the first register of the next instruction) if the address part of register n equals 1777. If it does not equal 1777, go to the next instruction in sequence. The Accumulator is unchanged.

OPR i n	500 + 20i + n	16 μsec*	OPR

OPERATE. The Operate instruction is a multi-purpose input-output instruction which is used to:

1) transfer information from the LINC keyboard and the control console toggle switches (Right Switches and Left Switches) to the LINC Accumulator.
2) control the transfer of digital information between the LINC and external digital devices.
3) provide pulses which may be used externally to synchronize or control special equipment.

Toggle Switch Input.

RSW. RIGHT SWITCHES. When n = 16, the contents of the Right Switches replace the contents of the Accumulator.

LSW. LEFT SWITCHES. When n = 17, the contents of the Left Switches replace the contents of the Accumulator.

In these cases the i-bit has no effect. *Time of Execution: 16 μsec.

Pausing. For $0 \leq n \leq 15$ the i-bit provides a timing control generally used to synchronize the LINC with external devices. When i = 1 the computer will pause. It will remain in an inactive state until it receives a "restart" signal (-3 volts) from the external equipment. The n-bits ($0 \leq n \leq 15$) of the instruction designate the external level input line to be used for restart. If i = 0, or if the restart signal is already present on line n, the computer will not pause. In this case it is assumed that the external equipment is ready for restart at the time the computer would normally pause.

Keyboard Input

KBD. KEYBOARD. When n = 15, the Accumulator is cleared and the 6-bit code number for a struck key is transferred into the right half of the Accumulator; the key is released. If i = 1, the computer waits for a key to be struck; if i = 0, or if a key was previously struck, the computer does not wait. When a key is struck, the keyboard locks until a KBD instruction is executed. *Time of Execution: 16 μsec when no pause.

Pulse Output. During the execution of **any** OPR instruction, four pulses of -3 volts are available to external equipment. The first of these is a long pulse which appears 4 μsec after the beginning of the instruction on one of 16 pulse output lines provided at the LINC's Data Terminal Box. The output line is designated by n ($0 \leq n \leq 17$); minimum duration of this pulse is 12 μsec. If the computer pauses, the pulse duration is extended by the length of the pause.

The other 3 pulses each .4 μsec duration, are associated with pause and restart. One is delivered to external equipment at pause time if, and only if, the computer actually pauses. The second occurs at the time when the computer would normally resume operation after a pause, regardless of whether the computer has actually paused or not. If the computer has paused, the pulse, which indicates that the computer is now running, will appear not less than 2 μsec and not more than 4 μsec after the restart signal has been delivered by the external equipment. If the computer has not paused, this pulse will appear 2 μsec after pause time. The third pulse appears 2 μsec later.

Digital Input-Output. The OPR instruction may be used to transfer 12-bit digital information between the LINC and external devices. The user may choose to transfer one word, into the LINC Accumulator each time the OPR instruction is executed, or he may use the instruction to transfer a group of words between the LINC memory and his external device. In this context the pause feature and the LINC's output pulses would be used to synchronize external equipment with the computer.

Digital Input to Accumulator. There are four 12-bit channels available for single-word input to the LINC Accumulator. Two, SN and TN, provide direct input to the Accumulator, and two, UN and VN, provide input via the B Register to the Accumulator. One word is transferred each time the Operate instruction is executed. When information is ready to be transferred, the external equipment must supply an enabling level for the appropriate channel (SNEL, TNEL, UNEL, or VNEL), followed, if the computer is paused, by the restart signal on line n. After restart, if the transfer is into the Accumulator via SN or TN, the Accumulator will be cleared automatically before the transfer takes place. Transfers into B via UN or VN are ORed (exclusive OR, partial add) with the contents of the Accumulator, and the result is left in the Accumulator. The Accumulator will not be cleared before UN and VN transfers unless a special clear enabling level (CLEL) is supplied along with the appropriate channel enabling level (UNEL or VNEL) before restart. *Time of Execution: 16 μsec. when no pause.

OPR

Digital Input to Memory. Information may be transferred from external equipment to the LINC memory via UN or VN. Information, transferred one word at a time, is stored in consecutive locations in the LINC memory. The number of words transferred each time the OPR instruction is executed is controlled by the external device. The computer will pause and transfer information repeatedly, until the external device indicates that no more transfers are to be made. Transfers are handled in the following way: the first word transferred must be a beginning address for storing subsequent transfers. This is transmitted over UN or VN, and the appropriate channel enabling level must be supplied (UNEL or VNEL). In addition, a begin transfer level (BEGT) and a memory input level (MINP) must be presented to the computer before restart. After restart, the computer transfers the word over UN or VN to the B register and to the memory address register. The Accumulator is cleared automatically, and the computer prepares to store subsequent transfers in the memory. The computer then pauses.

When the external device is ready with the first word of information, it must present enabling levels UNEL or VNEL and MINP before restart. The clear enabling level (CLEL) may be present. The begin transfer level (BEGT) may not be present. After restart the word is transferred to the B register over UN or VN, and stored in the LINC memory at the location specified by the memory address register. It is also ORed (exclusive OR, partial add) with the contents of the Accumulator and the result is left in the Accumulator. (The Accumulator will not have been cleared unless CLEL was present.) The contents of the memory address register are incremented by one in preparation for the next transfer, and the computer again pauses. The process is repeated as described above, until the last word is ready to be transferred. This time the external device presents only the channel enabling level (UNEL or VNEL) and the restart signal. MINP may not be present. After restart the computer completes the last transfer, and goes to p + 1 for the next instruction. The partial sum of all words transferred to the memory is left in the Accumulator.

*Time of Execution: 16 μsec. plus 8 μsec. for each word input to memory, exclusive of pauses.

OPR

External Output from Memory. Information may be transferred from the LINC memory to an external device directly from the B register. The operation is similar to memory input, except that a memory output level, MOUT, is presented instead of MINP. A beginning address must be supplied over UN or VN along with BEGT and MOUT, as described above; after restart the beginning address is transferred to the memory address register and the contents of the word at that location replace the contents of the B register. The computer then pauses. The external device completes the transfer from the B register. This time the MOUT level must be present before restart. CLEL is optional, and BEGT, UNEL, and VNEL may not be present. After restart the contents of the B register are ORed (exclusive or, partial add) with the contents of the Accumulator, and the result is left in the Accumulator. The contents of the memory address register are incremented by one, the next word in the memory replaces the contents of the B register, and the computer pauses. The process continues until the MOUT level is removed. The partial sum of all words transferred from the memory is left in the Accumulator, and the computer goes to p + 1 for the next instruction. The memory is left unchanged. *Time of Execution: 16 μsec. plus 8 μsec. for each word output, exclusive of pauses.

VII. SKIP CLASS INSTRUCTIONS

In these instructions the i-bit can be used to reverse the skip decision; that is, when i = 0 the computer will skip the next instruction (actually, the first register of the next instruction) only when the specified condition is met. However, when i = 1, the computer will skip only when the condition is not met; otherwise it will go on to the next instruction in sequence. The four situations which may arise are summarized in the following table in which p + **k**, **k** = 1 or 2, is the location of the next instruction.

TABLE IV. BRANCHING IN SKIP CLASS INSTRUCTIONS

i	condition	**k**	p + **k**
0	met	2	p + 2
0	$\overline{\text{met}}$	1	p + 1
1	met	1	p + 1
1	$\overline{\text{met}}$	2	p + 2

SNS i n	440 + 20i + n	8 μsec	SNS
SENSE SWITCH. Check to see if Sense Switch n (n = 0,1,..., 5 octal) on the control console is up (set to one), and go to p + **k** (see Table IV) for the next instruction.			

AZE i	450 + 20i	8 μsec	AZE
ACCUMULATOR ZERO. Check to see if the contents of the Accumulator equal positive or negative zero (all zeros or all ones) and go to p + **k** (see Table IV) for the next instruction. C(ACC) are unchanged.			

APO i	451 + 20i	8 μsec	APO
ACCUMULATOR POSITIVE. Check to see if the sign bit (bit 11) of the Accumulator is positive (zero) and go to p + **k** (see Table IV) for the next instruction. C(ACC) are unchanged.			

LZE i	452 + 20i	8 μsec	LZE
LINK ZERO. Check to see if the Link bit is zero and go to p + **k** (see Table IV) for the next instruction. C(ACC) and the Link bit are unchanged.			

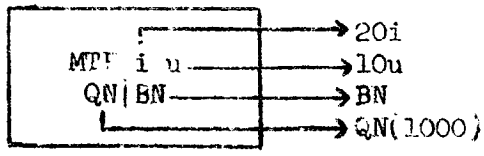
IBZ i	453 + 20i	8 μsec	IBZ
INTER BLOCK ZONE. Check to see whether either tape is in an Inter Block Zone and go to p + **k** (see Table IV) for the next instruction. A tape must be moving and up to speed for this condition to be met. The tapes are unaffected.			

SXL i n	400 +20i + n	8 μsec	SXL
SKIP ON EXTERNAL LEVEL. Check to see if external level n (n = 0,1,..., 14 octal) is present and go to p + **k** (see Table IV) for the next instruction.			

KST i	415 + 20i	8 μsec	KST
KEY STRUCK. Check to see if a key has been struck on the Keyboard and go to p + **k** (see Table IV) for the next instruction. The Keyboard is unaffected.			

VIII. MAGNETIC TAPE INSTRUCTIONS

MAGNETIC TAPE INSTRUCTION SUMMARY



i: Motion Control

i = 0 Tape stops

i = 1 Tape moves

u: Unit Select

u = 0 Unit #0

u = 1 Unit #1

QN: Quarter Number $0 \leq QN \leq 7$

QN	Memory Registers
0	0 - 377
1	400 - 777
2	1000 - 1377
3	1400 - 1777
4	2000 - 2377
5	2400 - 2777
6	3000 - 3377
7	3400 - 3777

BN: Block Number $0 \leq BN \leq 777$ (octal)

1 Tape = 512 (decimal) Blocks

1 Block = 256 (decimal) Words

1 Word = 12 (decimal) Bits

Data Sum = two's complement sum of 256 Words in Block

Check Sum = $\overline{\text{Data Sum}}$

Check Sum + Data Sum = Transfer Check

To Check: Transfer Check = -0

RDC i u	700 + 20i + 10u	RDC

READ AND CHECK. The specified Block is read into the specified Memory Quarter and the transfer is checked. If it does not check, the Block is read and checked again. If it checks, -0 is left in the Accumulator and the computer goes to p + 2 for the next instruction. The information on the tape is unchanged.

RCG i u	701 + 20i + 10u	RCG

READ AND CHECK GROUP. Consecutive Blocks are read into consecutive Memory Quarters and the transfers are checked. The BN bits in p + 1 specify the initial Block; bits 0-2 in p + 1 specify the initial Memory Quarter. Bits 9-11 in p + 1 (the QN bits) specify the number of additional Blocks to read after the initial Block. That is, C(bits 9-11) + 1 equal the total number of Blocks to read. If a Block does not check, it is read and checked again. When all Blocks have been read and checked, -0 is left in the Accumulator, and the computer goes to p + 2 for the next instruction. The information on the tape is unchanged.

RDE i u	702 + 20i + 10u	RDE

READ TAPE. The specified Block from the tape is read into the specified Memory Quarter, the Transfer Check is formed and left in the Accumulator. The computer goes to p + 2 for the next instruction. The information on the tape is unchanged.

MTB i u	703 + 20i + 10u	MTB

MOVE TOWARD BLOCK. The next Block Number, either forward or backward, is added to the BN bits (bits 0-8) of p + 1. (The QN bits are ignored.) The result is left in the Accumulator. If i = 1, the tape is left moving toward the Block specified by p + 1. If i = 0, the tape stops. The information on the tape and in memory is unchanged. The computer goes to p + 2 for the next instruction.

LINC ASSEMBLY PROGRAM 3

LAP3

The following is a description of LAP3, a symbolic conversion program for the LINC which uses the LINC keyboard for on-line input, and the magnetic tapes for storage, working area, and output. The scope is used to provide readable manuscript.

I. General Information

A. Operating Procedure

1. LAP3 occupies Blocks 300 - 327 of the tape, plus Blocks 330 and following for "working area" (see Chart IV). The tape must be on Unit 0.

2. To operate, READ BLOCK 300 into QUARTER 0, and start at 0. A "0001" will appear on the scope to indicate that LAP3 is ready to accept keyboard input. This is the only start or restart procedure.

3. Lines of manuscript and meta commands are typed into the computer via the keyboard. LAP3 displays on the scope the information being typed, one line at a time, as it is keyed in.

4. One quarter of the LINC memory is used to collect manuscript. As the quarter is filled, it is saved on the tape beginning in Block 336 of the working area. It takes approximately 100 - 110 (octal) lines of manuscript to fill one block.

B. Manuscript Lines

1. By "manuscript line" is meant a line of program, a comment, an origin, or an equality. A meta command is not a line of manuscript.

2. A manuscript line may not exceed 17_{10} struck characters. This includes spaces, case shifts, and the terminator.

3. Manuscript lines are always terminated by striking the "End-of-Line" key, EOL. Lines which are too long are automatically terminated with an EOL by LAP3 and called to the attention of the typist. (See Error Detection.)

C. Manuscript Line Numbers

1. LAP3 assigns a "line number" to every line entered. The numbering appears at the upper left of the line on the scope; it is sequential, beginning with 1, and octal.

2. The number "1" appears as the first line number when the initial "start 0" is executed. After that a new line number appears every time the EOL is struck in terminating a manuscript line, and the computer waits for the next line to be typed.

D. Deleting

1. Hitting the delete key, del, will delete the current line. If there is no current line (i.e. if the computer is displaying only a line number), the previous line will be deleted. In either case, the line preceding the deleted line will appear on the scope.

Example: The following sequence will appear on the scope one line at a time as it is typed:

```
0001
      ADD 3       ---hit EOL---
0002
      STC 5       ---hit EOL---
0003
      JMP 56      ---hit EOL---
0004
      STA         ---hit "del;" line 4 is deleted---
0003
      JMP 56      ---line 3 reappears; type line 4 again---
0004
      STC 10      ---hit EOL---
0005
                  ---next line number appears; hit "del"---
0003
      JMP 56      ---line 4 is deleted, line 3 reappears; hit "del"
0002
      STC 5       ---line 3 is deleted, line 2 reappears; continue
```

2. Whatever is "deleted" is permanently deleted from the manuscript. Whatever is displayed on the scope is the most recent line recorded. In the example above, only lines 1 and 2 are still part of the manuscript at the end of the sequence.

3. It is not necessary after a delete to hit EOL before continuing with the next line.

E. Display Format

1. The display format is as in the above example, one line at a time.

2. All keyboard characters (see Chart II) are displayed except EOL, del, CASE, and META.

3. Characters are displayed as they are struck.

F. Case Shift

1. Some keys on the keyboard have been assigned both upper and lower cases. The characters in the middle of the keys are lower case (see Chart II), and LAP3 normally interprets the keyboard as lower case.

2. To select upper case, hit the case shift key, CASE, and then hit the upper case character.

3. The shift is not permanent; it is good for <u>one character only</u>. LAP3 returns to lower case automatically.

4. LAP3 will interrupt the display after CASE is struck, until the upper case character is also struck. When both have been struck, the display will resume.

II. Line Format and Symbols

A. Origins

1. Origins must be specified as octal constants, preceded by an origin character, ⊟, and terminated with an EOL.

Example: to specify an origin of 300 on line 1, type

1

⊟ 300_{EOL}

2. Spaces are not permitted anywhere on an origin line except before the origin character.

3. An origin line may not contain a line of program. If a line of program appears before an origin character on the same line, the program line will be omitted during conversion. If it appears after, the origin will be interpreted incorrectly.

4. Origins may be specified throughout the program. If portions of the binary overlap as a result, the later origins take precedence.

5. Conversion is faster if origins referring to the same quarter of memory are entered consecutively in the manuscript, i.e. not interspersed with origins referring to different quarters. This technique is not required, but it saves much tape shuffling during conversion.

6. Programs with no initial origin will be located at 20.

B. Comments

1. Comments are permitted anywhere in the manuscript so long as they occupy a line by themselves.

2. A comment line must begin with the comment character (|). It may not begin with a space.

3. If a comment is included on a line of program, the program line will be omitted during conversion.

C. Tags

1. Any program line (i.e. any manuscript line except origins, comments, and equalities) may be tagged. That is, it may be identified by a symbol which, during conversion, will correspond to the actual memory location of the program line.

2. A tagged line must begin with the tag symbol, #. It may not begin with a space.

3. Tags are limited to two characters.
   a. They must be of the format "number, letter."
   b. The numbers are 1 through 7; letters are capitals, A through Z.

c. No spaces are permitted within the tag.
e.g. #2A is correct; # 2A is not.

d. Any other format, or any other combination of characters will be called to the typist's attention. (See Error Detection)

4. No tag terminator is required.

5. A "number, letter" combination may be used once as a tag (#) if it is not also used as an equality (=).

D. Symbolic Operation Mnemonics

1. All first-order three letter mnemonics for operation codes are permitted. Substitute mnemonics as defined by Chart I are also permitted.

2. The mnemonics used must agree with Chart I, except that the second character may be replaced by any other capital letter.
e.g. JMP or JBP are both acceptable for JUMP.
JiP is not.

3. No spaces are permitted within the mnemonic.

E. Special Symbols

1. Bit 4 (the i-bit) is specified symbolically with "i." Typing "i" on a program line will cause bit 4 to be set to a 1 during conversion.

2. Bit 3 (the tape unit bit) is specified symbolically with "u." Typing "u" on a program line will cause bit 3 to be set to a 1 during conversion.

3. The vertical bar (|) is used in the second line of tape instructions to separate QN and BN.
e.g. QN = 3 and BN = 45, written "3|45," will be converted to 3045.

a. Spaces are permitted as indicated by apostrophes:
'3'|'45'

4. "Present Location" is specified symbolically with "p."

5. The "+" and "-" symbols are used as "plus" and "minus" in relative addressing and in assigning the sign of a number. The "-"

symbol is also interpreted as "dash" for some of the meta commands.

e.g. -567 will be converted to 7210.

6. The equality symbol, =, is used to assign a value to an undefined "number, letter" combination. It is not a tag and it may not be used to assign a location to a tagged line.
   a. Equalities are permitted anywhere in the manuscript so long as they occupy a line by themselves.
   b. The numbers are 1 through 7; letters are capitals, A through Z.
   c. The "number, letter" combination must be on the left of the "=" symbol; the numerical assignment on the right.
   d. No spaces are permitted anywhere on an equality line.
   e. The numerical assignment may not be signed.
      e.g. 6G=7774 is legal;
      6G=-3 is not.
   f. A "number, letter" combination may be defined once by an equality (=) if it is not also defined by a tag (#). If a "number, letter" combination is defined more than once, the last definition entered in the manuscript will be the one used during conversion.

F. Numerals

1. Numerals on any line of manuscript or in any meta command must be octal constants. LAP3 will convert incorrectly any numbers containing an 8 or 9.

2. Spaces are not permitted between the digits of a number.
   e.g. 7745 is legal; 774 5 is not.

G. Address Field

1. Symbolic and relative addressing with any combination of "number, letter," numerals or "p" is permitted.
   e.g. JMP p-5
   ADD 6+4K
   3C+6-p
   -4+7Z

2. No spaces are permitted within the address field.

3. Undefined "number, letter" combinations in the address field are assigned the value zero.

   e.g. JMP 3X, when 3X is not defined, will be converted to 6000.

4. For multiply defined "number, letter" combinations, LAP3 will use the last one entered in the manuscript for the assignment, regardless of whether it was entered with # or =.

   e.g. The following will be converted as shown:

Location	Manuscript	Conversion
	⊟ 100	
100)	#2F ADD 3	2003
101)	JMP 2F+2	6042 (NOT 6102)
	2F=40	

II. Spacing

1. No spaces are required anywhere in a line except as desired by the typist for scope placement.

2. Spaces may not be inserted:
   a. Within tags: (#2 K).
   b. Within origins: (⊟ 270).
   c. Within symbolic operation mnemonics: (S T").
   d. Between the digits of a number: (3 45).
   e. Within the address field: (3X- 5).
   f. Within equalities: (4L =770).

3. Spaces may be inserted between the tag, operation, index, address and vertical bar fields of the line.

   e.g. #3D STA i 3X
   RDC i u
   2 | 100

4. Lines which begin with either a tag (#) or a comment ([) symbol are automatically positioned at the left of the scope. All other lines will appear toward the middle of the scope.

I. Error Detection

1. Some lines which contain errors are detected by the compiler while they are being keyed in. These are:

a. Tagged lines which begin with an illegal tag or tag format.
b. Lines which are too long.
c. Lines which contain either a tag or an origin character anywhere except first on the line. This includes comments.

2. Faulty lines are held on the scope and the keyboard is briefly locked. When the keyboard releases, the typist must hit either:
"del," which will delete the line, or
"G," for (GO) which will accept the line as usual.
The keyboard will repeatedly lock until one of these keys has been struck.

3. Error detection for faulty meta commands is somewhat different, and will be described below.

J. Line Format

1. Program Lines

a. The following formats for program lines are permissable (spaces optional):

```
#3K  SAE  i  5
#3K  i  SAE  5
#3K  SAE  5  i
#3K  5  i  SAE
#7S  RDC  u i
```

b. The formats

```
#3K  5  SAE  i
#3K  i5  SAE
#3K  3X+5  STC
```

are permitted, but the space is required between the "5" and the "S."

c. "i" and "u" may come anywhere on a line.

d. Generally, any format is acceptable, so long as
1) Tags come first.
2) Numbers and letters are distinguished from "number, letter" combinations. (Examples in b. above.)

III. Meta Commands

A. LAP3 provides nine meta commands for changing, controlling, and converting manuscript.

1. Except for the terminator, meta commands are entered exactly as regular manuscript lines. They are displayed with a line number and may be deleted with "del" any time before the terminator is struck.

2. The meta command terminator is a Case Shift (CASE) followed by the EOL key; this combination is marked META on the keyboard.

3. Meta commands are executed <u>when they are entered</u>, and automatically deleted from the manuscript at that time. After a meta command is executed, LAP3 returns to normal input operation, displaying the current line number on the scope. Continue typing.

B. Errors: there are two kinds:

1. When a faulty meta command is entered with the META key, a question mark (?) will appear on the scope following the command, and the keyboard will lock briefly. The only key LAP3 will accept at this point is "del." Try again.

2. Once the command has been accepted, if LAP3 finds that it cannot then be executed, a "NO" will appear on the scope. (This does not happen until the tapes have churned a while and LAP3 has at least tried.) The "NO" will remain on the scope until a key (any key) is struck; LAP3 will return to normal input operation displaying the current line number. The manuscript entered up to this point is still intact (except with READ MANUSCRIPT).

C. Formats

1. Meta commands must be at least two letters as specified on Chart III. If no numeric parameters follow, any other characters may also be typed: e.g., "Display" may be specified with <u>DI</u>, <u>DIS</u>, <u>DISP</u>, <u>DISPLAY</u>, etc., so long as the "DI" is present. However, when numeric parameters follow, only two letters are permitted.

2. <u>Spaces are only permitted between the command and the parameters</u>.

D. Except for "Read Manuscript," "Copy," and "Convert Manuscripts," commands are effective <u>only</u> for manuscript <u>in the working area of the tape</u> (see p. 1, I-A).

E. Commands

1. REMOVE

Format: RE LN,n_{META}
or: RE LN-LN+n_{META}

a. Lines may be removed from the manuscript by typing "RE" followed by the line number, LN, (spaces optional) of the first line to be removed. This is followed by a comma and the number of lines (octal) to remove, or by a "-" and the first line number after the area to be removed.

Example: To remove 5 lines beginning with line 230, when the manuscript presently goes through line 402, type (on line 0403):

0403
RE 230,5$_{META}$ or 0403 RE 230-235$_{META}$

When the META character is entered, LAP3 will execute the command; the rest of the manuscript is automatically renumbered, and LAP3 returns to normal input operation by displaying a 0376 as the next line number.

b. When a REMOVE includes the last line in the manuscript, any terminating parameter beyond that point will suffice to remove the lines.

Example: To remove the last 10 lines of a manuscript which presently ends at line 164, type

0165
RE 155,10 (or any number greater than 10)$_{META}$

or 0165 RE 155-165 (or any number greater than 165)$_{META}$

LAP3 will return with 0155 on the scope as the next line number.

c. LAP3 will respond with a "NO" when a REMOVE requests a line number (as the initial parameter) not contained in the manuscript. Example: REMOVE line 20, when the manuscript only goes through line 10.

2. INSERT
END META

Format: IN LN$_{META}$
EN$_{META}$

a. Lines may be inserted in the manuscript by typing "IN," followed by the line number of the line following the place the inserts are to be put. "IN 30_{META}" means "insert the following before line 30." A 0030 will appear on the scope as the next line number; lines to be inserted are entered at this point just as regular lines. They may be deleted with "del," just as regular lines, but LAP3 will delete only through line 0030. When all the lines have been entered, type "EN_{META}" (as a separate line). LAP3 will make the inserts and return with the new present line number on the scope.

Example: If 3 lines are to be inserted before line 40 in a manuscript which is presently 105 lines long, the following sequence will appear on the scope (one line at a time):

0106	
IN 40_{META}	Type meta command
0040	
.....EOL	
0041	
.....EOL	Enter the 3 lines
0042	
.....EOL	
0043	
END_{META}	End meta command
0111	New line number appears; continue typing.

The commands on lines 106 and 43 are deleted automatically when they are executed.

b. Following the IN command but preceding the EN command, LAP3 will accept no other **meta** commands.

c. LAP3 permits the user to insert up to 2 memory quarters of of information with one INSERT command, i.e. approximately 200 - 220 lines (see Chart IV). Should this much be inserted without terminating the command, it will be automatically terminated by LAP3, the inserts will be made, and the new line number will appear on the scope. The user may continue inserting by giving a new INSERT command.

d. LAP3 will respond with a "NO" when an INSERT requests a line number not contained in the manuscript. Example: INSERT before line 50, when the manuscript only goes through line 42.

3. PACK Format: PA_{META}

The meta commands INSERT and REMOVE leave gaps in the manuscript wherever the change is made. When several of these commands are executed, the number of tape blocks occupied by the manuscript can become quite large; the length of time required to execute further commands grows proportionately. PACK will condense the manuscript; it does not, however, change it in any other way. Giving a PACK command when no INSERT or REMOVE has been executed does nothing (except to make the tapes churn).

4. DISPLAY Format: DI_{META}

a. This command will display from 1 to 70 (octal) lines of manuscript on the scope, under knob control by the user.

1) Knob 0 is used to select the number of lines per frame. A request for 20 or fewer (octal) lines will produce a single column display of readable size characters. Requests for between 21 and 70 lines will be displayed in two columns of small size characters.

2) Knob 1 is used to sweep through the manuscript, advancing the frame one line at a time either forward or backward. The middle positions on this knob will hold the display stationary. Hitting the CASE key when this knob is on stationary will advance the display by one frame (e.g. for taking pictures).

b. Lines are displayed with line numbers.

c. To terminate the display, hit EOL. LAP3 will return to normal input operations.

d. If a request is given to DISPLAY an unpacked manuscript, LAP3 will PACK it automatically before displaying it.

5. SAVE MANUSCRIPT Format: SM_{META}

a. Manuscript in the working area of the LAP3 tape can be saved at any time in any consecutive blocks on either unit. Saving

manuscript more frequently than you think necessary is recommended. Saving manuscript in the LAP3 area on the tape is not recommended.

b. An unpacked manuscript is automatically packed before the SM command is executed.

c. When the user types "SM_{META}", the following appears on the scope:

```
SAVE
n BLOCKS
AT BLOCK ?,
UNIT ?
```

1) "n," supplied by LAP3, is the number of blocks occupied by the manuscript; because of a control block which accompanies every manuscript, n is never less than 2.

2) The question marks are to be filled in by the user: type the block number of the first block where you want the manuscript to be put. This will replace the "?" on line 3 above. Terminate the line with EOL and type the unit number. This will replace the "?" on line 4 above. Terminate finally with a second EOL and the command will be executed.

3) If you don't like what you typed, hit "del" and the question mark(s) will reappear. One "del" restores one "?". Type the entry again. (Do any "del"s before the final EOL.) If LAP3 doesn't like what you typed, the question mark(s) will reappear automatically when the EOLs are struck.

d. After the SAVE, LAP3 will return to normal input operation; the manuscript will be as it was before the command was executed (except that it may have been packed).

6. READ MANUSCRIPT Format: RM BN,UN_{META}

READ MANUSCRIPT will restore a manuscript which has been SAVED to the working area of the LAP3 tape. Type "RM" followed by the block number (BN), a comma, and the unit number (UN) specifying the present location of the manuscript.

a. Only manuscripts which have been saved with a SAVE MANUSCRIPT can be read with a READ MANUSCRIPT.

b. When the command is executed, LAP3 will return with the present line number for the manuscript just read. Continue typing.

c. LAP3 will return with a "NO" when either the BN or the UN requested is illegal, or when there is no SAVED manuscript at that location. Hitting a key (any key) at this point will return LAP3 to normal input operation, but it will return with a "1" as the present line number. Whatever you were typing up to the READ command is effectively lost.

7. CONVERT Format: CV_{META}

The "CV" command converts to binary the manuscript in the working area of the tape.

a. The binary version will be in blocks 330 - 333 of the tape on unit 0 (the LAP3 tape) after conversion. Block numbers correspond to memory quarters 0 - 3 respectively.

b. After conversion LAP3 will return to normal input operation; the manuscript will be as it was before conversion.

8. CONVERT MANUSCRIPTS Format: CM_{META}

To convert a manuscript NOT in the working area of the tape, or to convert several manuscripts together, the command "CM" is used. After the command is given, the following will appear on the scope:

```
CONVERT
MANUSCRIPTS AT
```

a. Type the block number(s) specifying the location of the beginning of each manuscript to be converted. Separate the block number entries with spaces. The numbers will appear on the scope as they are typed; "del" will delete them one at a time. LAP3 will delete non-existent block numbers when they are typed.

b. The manuscript(s) specified must all be on the tape on unit 0.

c. Multiple manuscripts are converted together in the order in which they are requested; i.e. they are treated as one longer manuscript. (This has relevance to origins in the manuscripts.)

d. The manuscript(s) may NOT be in the working area of the tape. Only manuscript(s) which have been saved with a SAVE MANUSCRIPT may be converted with CM.

e. As many as 8 manuscripts may be selected. When 8 have been selected, LAP3 will terminate the selection automatically and execute the command. Otherwise:

f. Terminate the manuscript selection with EOL. LAP3 will convert the manuscript(s) and return to normal input operation. The manuscript in the working area will be as it was before the CM command was given.

g. As with CV, the binary conversion will be in blocks 330 - 353 on unit 0, block numbers corresponding to memory quarters 0 - 3, respectively.

h. A "NO" will appear if LAP3 finds that any of the blocks specified does not contain the beginning of a SAVED manuscript.

9. COPY Format: CP_{META}

This command permits the user to copy any number of blocks to any place on either unit. (It does not apply only to manuscript. When the command is given, the following appears on the scope.

```
COPY
? BLOCKS          <-- Number of blocks to move
FROM BLOCK ?  }   Present location
UNIT ?        }
TO BLOCK ?    }   Requested location
UNIT ?        }
```

a. Fill in the question marks as indicated, terminating each line entry with EOL. The command will be executed when the last EOL is typed after line 6 above. Hitting "del" will delete entries, one at a time. LAP3 will delete illegal entries automatically.

b. Since LAP3 can only copy 3 blocks at a time (because of memory limitations), care should be taken not to overlap the block numbers when requesting a COPY. Example: Requesting a COPY of 6 blocks from block 550 to block 555 on the same unit will not work. Requesting a COPY of 3 blocks, however, from block 550 to block 551 on the same unit will work. (Obviously, if the units are different the COPY will be successful.)

c. After the COPY, LAP3 will return to normal input operation; the manuscript in the working area will be as it was before the command.

M. A. WILKES

August 8, 1963

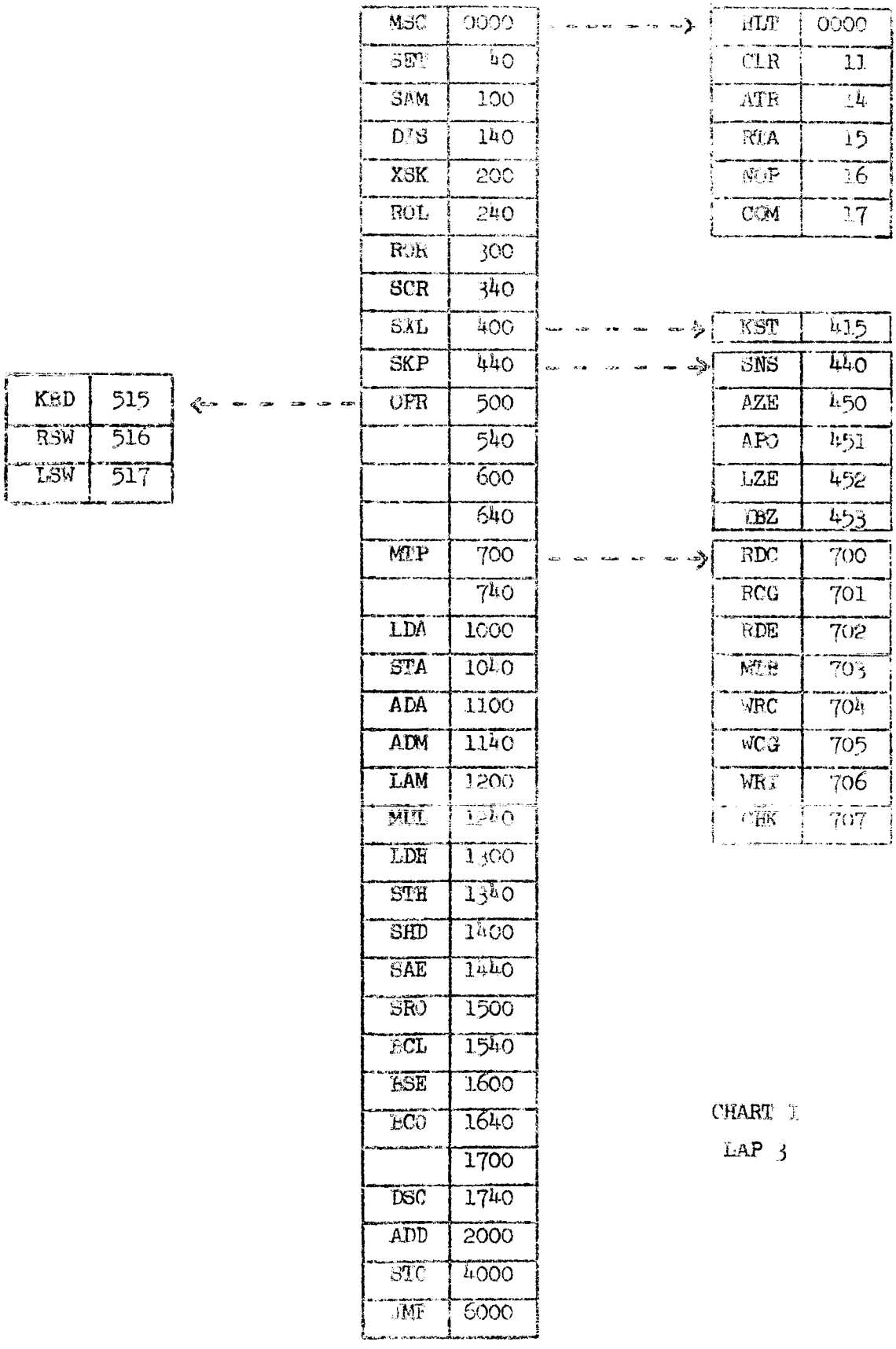
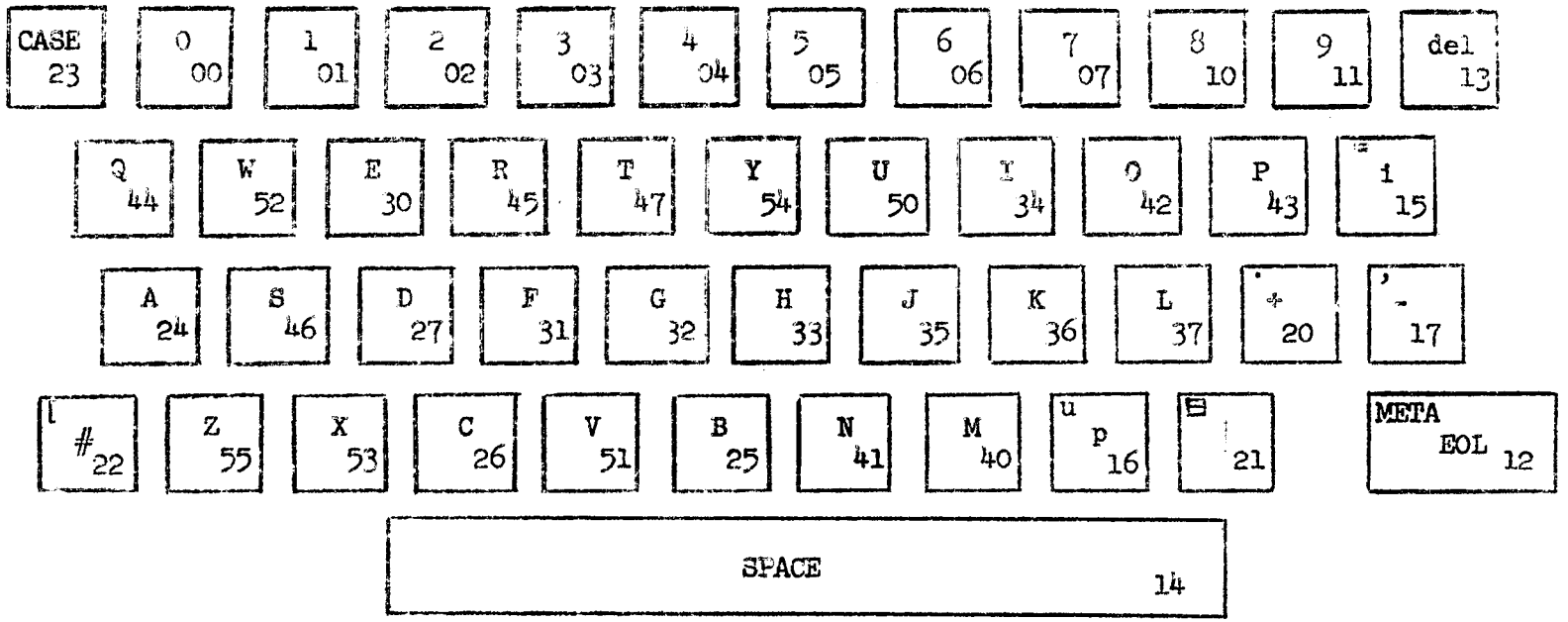

MSC 0000
SET 40
SAM 100
DAS 140
XSK 200
ROL 240
ROR 300
SCR 340
SXL 400
SKP 440
OPR 500
540
600
640
MTP 700
740
LDA 1000
STA 1040
ADA 1100
ADM 1140
LAM 1200
MUL 1240
LDH 1300
STH 1340
SHD 1400
SAE 1440
SRO 1500
ECL 1540
ESE 1600
ECO 1640
1700
DSC 1740
ADD 2000
STC 4000
JMP 6000
HLT 0000
CLR 11
ATR 14
RIA 15
NOP 16
COM 17
KST 415
SNS 440
AZE 450
APO 451
LZE 452
OBZ 453
RDC 700
RCG 701
RDE 702
MZE 703
WRC 704
WCG 705
WRT 706
CHK 707
KBD 515
RSW 516
LSW 517

CHART I

LAP 3



|	Comment	CASE	Case Shift
#	Tag	i	i-bit
⊟	Origin	=	Equality
u	Tape Unit	del	Delete
EOL	End of Line	¦	QN¦BN
p	Present Location		

CHART II

LAP 3

Chart III. LAP3 Meta Commands

Command	Required Format	May be followed by other characters
1. Remove	RE LN,n RE LN-LN	No
2. Insert	IN LN	
End Meta	EN	Yes
3. Pack	PA	
4. Display	DI	
5. Save Manuscript	SM	
6. Read Manuscript	RM SN,ON	No
7. Convert	CV	Yes
8. Convert Manuscripts	CM	
9. Copy	CP	

Chart IV. LAP3 Tape Allocation

Block	Allocation
300-322	LAP3
323	Temporary Storage
324-325	Inserts
326-327	LAP3
330-333	Binary
334	Temporary Storage
335	Manuscript Control Block
336 and ff.	Manuscript

Chart V. LAP's Tape Allocation

Block	Allocation
300	Regular Input
301	COPY Meta Command
302	READ MANUSCRIPT Meta Command
303 304 305 306	CV and CM Meta Commands and Conversion control routines
307	(not used)
310	SAVE MANUSCRIPT Meta Command
311 312	DISPLAY Meta Command
313	PACK Meta Command
314 315	INSERT Meta Command
316	(not used)
317	REMOVE Meta Command
320	(not used)
321 322	Regular Input
323	Temporary Storage
324 325	Temporary Storage for INSERTED lines
326	Pass I for Conversion
327	Pass II for Conversion
330 331 332 333	Binary Version after Conversion
334	Temporary Storage
335	Manuscript Control Block
336	Manuscript
↓	↓

LINC Control Console

Located on the control console are most of the elements required for manual control of the computer. The toggle switches, arranged in groups of threes, present digital information to the computer. The pushbuttons, levers, and rotary switches initiate computer actions and special modes of operation. In addition, indicator lights show the state of the major computer elements.

Register Lights

The register lights display the contents of the Accumulator and various control registers; they are arranged in groups of threes to facilitate reading in octal form. A light which is lit corresponds to a flip-flop in the "on" or "one" state. The rightmost light corresponds to bit 0 of the register.

INSTRUCTION - 12 bits. The Instruction lights display the code number of the instruction currently being executed by the computer. During the execution of DSC, MUL, ROR, ROL and SCR, the rightmost four bits of the Instruction register are used for counting. Otherwise, the Instruction lights change only when the computer is ready to execute a new instruction.

INSTRUCTION LOCATION - 10 bits. During cycle 0 (the 1 cycle) of all instructions withdrawn from the memory for execution, the Instruction Location lights display the memory location of the current instruction. In later cycles, except for MTP, XSK, SHD, and SAE, these lights display the location of the instruction the computer will execute next. In later cycles of MTP the lights display the location of the second word of the MTP instruction; for XSK, SHD, and SAE, they display the location of the next instruction in sequence, regardless of whether that instruction may be skipped or not. An instruction executed manually from the console with the DO TOG INSTR lever affects the Instruction Location lights only when the instruction itself is JMP.

ACCUMULATOR and L - 13 bits. The Accumulator (12 bits) and the Link bit (marked L) are affected only by executing instructions which require their use. They cannot be disturbed by any console operations except DO [illegible] INSTR.

MEMORY CONTENTS - 12 bits. These lights display all words withdrawn from or stored in the memory. They also display the multiplicand during the last part of a MUL instruction.

MEMORY ADDRESS - 11 bits. The Memory Address lights display the memory location of any word withdrawn from or stored in the memory. Except for later cycles of MUL, DIS, SAM, OPR, and the shift instructions, these lights display the location of the word in the Memory Contents register.

RELAYS. The six Relay lights display the state of the relay flip-flops. They are set to zero when power is turned ON or OFF. Otherwise they are affected only by the ATR instruction.

RUN Lights

Two lights marked R (Run) and P (Pause) indicate the state of the computer's activity. When the computer is operating under program control, the Run light is normally lit. The Run light also comes on during most pushbutton operations. The Pause light will come on during MTP instructions and when a pause is initiated during OPR instructions. When neither light is lit the computer is STOPPED.

CYCLE Lights

The lights marked I, X, O, and E display the cycle periods of an instruction as it is executed by the computer. The I light corresponds to Cycle 0, or the instruction cycle, during which the instruction is

brought into the Instruction register. Cycle 1, marked X, is the index cycle, so called because all instructions except XSK which index a word in the β registers do so in Cycle 1. O and E, the operation and execution cycles, correspond respectively to Cycles 2 and 3.

Toggle Switches

Three sets of toggle switches are available for presenting information to the computer. A switch which is up is said to be in the "one" state. As with the register lights, the rightmost switch is switch 0. The SENSE SWITCHES (6 toggle switches) can be sensed, one switch at a time, by the SNS instruction. They are used while a program is running to control program branching. The RIGHT and LEFT SWITCHES (12 toggle switches each) can be read into the Accumulator with the RSW and LSW instructions. These switches are also used extensively in conjunction with the pushbuttons and levers as described below.

Pushbuttons, Levers, and Rotary Switches

Control of the computer is provided through the various pushbuttons and levers at the control console. Some execute instructions or monitor stored instructions as they are executed by the computer. Others are used to examine or change the contents of words in the computer memory, and others, such as MARK and CLEAR, change the operating state of the computer altogether.

The pushbuttons labelled AUTO RESTART, ISTOP, XOESTOP, FILL, EXAM, CYCLE BY CYCLE, INSTR BY INSTR, CLEAR, and MARK will light when pushed. The other pushbuttons and the levers do not light. Generally a lighted pushbutton indicates that the computer is operating in the stated mode. Those buttons with no light initiate single actions; to repeat the action the button or lever must be pressed again.

Most of the pushbuttons and levers are mutually exclusive; operating one of them will generally disable the others. Pushing a button which is illuminated will simply repeat the indicated action. A mode is generally left only by initiating some other console action leading to a new mode. Exceptions to this are described individually below.

The control console is designed so that most actions can be initiated at any time without damaging a running program or a tape transfer. When an action is initiated while the computer is Running (the R light on), the computer waits until a "safe" time (t_1 time of the I Cycle (0) of the next instruction) before changing mode. That is, it will always finish an instruction whose execution is in progress before changing to a new mode. The same is true if an action is initiated when the computer is Stopped in the X, 0, or E Cycles (Cycles 1, 2, or 3). Under these circumstances the computer will be set to Run and will proceed to the next "safe" time, t_1 time, of the I Cycle of the next instruction. (The exception to this occurs in STEP actions when CYCLE BY CYCLE is on; see below.) Furthermore, console actions do not affect the Instruction Location register or the Accumulator (unless they are specifically intended to do so, e.g. START 20). The user, therefore, can always resume normal running operation after interrupting a Running or Stopped computer.

The Stop lever is the only means of interrupting the computer if the computer is in the Pause state. If the computer is interrupted while Paused, it will not finish the instruction.

The above remarks about mode changing do not pertain to the AUTO RESTART, ISTOP, and XOESTOP pushbuttons. These take effect independently of the rest of the console, and are recognized without interruption of the computer.

ON. The ON pushbutton supplies power to the computer and initiates certain "presetting" actions which turn control elements to an "off" or "safe" state. Pushing this button when the computer is off will start the central clock and leave the computer in a ready state to begin operations.[1] The Run, Cycle, Instruction, and Relay lights will be cleared, the console pushbuttons cleared, the tape write-gate shut off, and the computer and the tapes left in the STOPPED state. The contents of the memory and the tapes are not affected. Pushing ON when the computer is already on has no effect.

OFF. The OFF pushbutton removes power from the computer. As power disappears, presetting pulses appear temporarily as in turning ON. However, since this presetting is not at present synchronized with other actions, it is advisable to make sure the computer is STOPPED (the RUN lights both out) before pushing OFF. Pushing OFF when the computer is running (R) or paused (P) may destroy a word in the memory or on the tape. Pushing OFF when the computer is already off has no effect.

CLEAR. The red CLEAR pushbutton clears the computer memory and does a simple memory read-write test. Pushing this button clears all other console functions and puts the computer in a clear and test mode; the CLEAR light and the R light will come on. The Cycle lights will go out. On alternate passes through the memory the computer writes and reads zeroes, then writes and reads ones, in each memory register. Should the memory test fail, the computer will STOP (RUN lights out). The Memory Address register and the Memory Contents register will display the location and contents of the memory register which failed. If the computer was writing and reading zeroes at the time of the failure, all the bits in the Memory Contents register will be zero except the bit(s) which failed, indicating that the memory has "picked up a one." If the computer was writing and reading ones, all the bits in the Memory Contents register will be ones, except the bit(s) which failed, indicating that the memory has "dropped a one." The computer will stay in the CLEAR mode until some other console

1 No warm-up time is necessary except for the scopes, which require about 40 secs.

action is initiated. Regardless of whether the computer was on a pass testing for zeroes or testing for ones, the entire memory is always left with zeroes when the CLEAR mode is interrupted. ISTOP and XOESTOP are not recognized when the computer is in the CLEAR mode.

MARK. The red MARK pushbutton is used in conjunction with a special program to generate LINC tapes. Pushing this button clears all other console functions except ISTOP and XOESTOP, puts the computer into the MARK TAPE mode, and begins executing instructions at location 40. Although the computer operates under program control when generating tapes, the SAE instruction behaves differently in the MARK mode. Although some console actions may be initiated, they will disrupt the tape generation timing and may disable the MARK mode itself. Therefore no other console actions should be initiated while the MARK light is on. Likewise, the MARK button should not be pushed except when it is expressly desired to generate a tape. However, unless the computer encounters an MSC 13 instruction, pushing the MARK button will not damage the information on tape. The MARK button can be turned off by starting some other console function, or by executing a HLT or MTP instruction.

EXAM. The EXAM pushbutton is used in conjunction with the Left Switches to examine one memory word. Pushing this button interrupts any operation the computer might have been carrying out, and clears all other console functions except ISTOP and XOESTOP. When EXAM is pushed, the EXAM light will come on and the contents of the word whose address matches the setting of the Left Switches will appear in the Memory Contents lights. The setting of the Left Switches will appear in the Memory Address lights. The computer will STOP (RUN lights out). It will remain in the EXAM mode until some other console action is specified.

FILL. The FILL pushbutton is used in conjunction with the Left and Right Switches to fill one memory word from the console. Pushing FILL interrupts any operation the computer might have been carrying out, and clears all other console functions except ISTOP and XOESTOP. When the FILL button is pushed, the FILL light comes on and the word held in the Right Switches is stored in the memory at the location specified by the Left Switches. The settings of the Left and Right Switches will appear respectively in the Memory Address and Memory Contents lights. When the FILL button is released, the light goes out and the EXAM light comes on. The computer examines the word addressed by the Left Switches, just as though EXAM had been pushed, and STOPS. The computer will remain in the EXAM mode until some other console function is specified. Pushing FILL affects only the memory word addressed by the Left Switches; it may be used when the computer is Running or Stopped.

STEP EXAM. The STEP EXAM lever interrupts any operation the computer might have been carrying out, and clears all other console functions except ISTOP and XOESTOP. It operates like EXAM except that the Left Switches are not read into the Memory Address register. Instead, STEP EXAM steps the contents of the Memory Address register by 1, and displays in the Memory Contents register the word at that location. The EXAM light comes on, and the computer STOPS. It will remain in the STEP EXAM mode with the EXAM light on until another console action is initiated. STEP EXAM may be used when the computer is Running or Stopped.

FILL STEP. The FILL STEP lever interrupts any operation the computer might have been carrying out, and clears all other console functions except ISTOP and XOESTOP. It operates like FILL except that the Left Switches are not read into the Memory Address register. Instead, when FILL STEP is pressed, the FILL light comes on and the contents of the Right Switches are read into the Memory Contents register and are stored in the memory at the location specified by the Memory Address register. When FILL STEP is released, the computer behaves as in STEP EXAM and Stops. That is, the contents of the Memory Address register are stepped by 1, and the word

at that location is displayed in the Memory Contents register. The EXAM light comes on and the computer is left in the STEP EXAM mode. FILL STEP may be used when the computer is Running or Stopped.

Selective Starts

The following console actions interrupt any operation the computer might have been carrying out, and cause it to start executing instructions at the selected location. They clear all other console functions except ISTOP and XOESTOP, and put the computer directly into the Run state. They may be used when the computer is Running or Stopped.

START 20. The computer begins executing instructions starting at location 20.

START 400. The computer begins executing instructions starting at location 400.

START RS. The computer begins executing instructions starting at the location specified by the Right Switches (RS).

RESUME. The computer begins executing instructions starting at the location presently displayed in the Instruction Location lights. It can therefore be used to resume normal running at full speed after an interruption.

Selective Stops

The LINC can be made to Stop under specified conditions in the course of running a program. The following stops are especially useful when trying to detect program errors. The LINC always stops at t_1 time (see Instruction Timing Diagrams), regardless of the kind of Stop specified. Any of the following stops may be specified when the computer is Running or Stopped.

INSTR BY INSTR and CYCLE BY CYCLE. These pushbuttons interrupt any operation the computer might have been carrying out and clear all other console functions except ISTOP and XOESTOP. Pushing either button when the computer is Running will cause it to Stop at the next I Cycle, t_1 time, i.e. at the beginning of the next instruction. Pushing either button when the computer is Stopped in the I Cycle has no effect except to clear other console functions. Pushing either button when the computer is Stopped in any cycle other than the I Cycle will cause the computer to Run until the next I Cycle, t_1 time and Stop at the beginning of that instruction. The pushbutton light comes on and the computer is left ready to execute one instruction, or one cycle of an instruction, every time the STEP lever is pressed. When operating in either of these modes, the computer executes stored programmed instructions just as it does when Running at full speed, except that it Stops at the specified points.

INSTR BY INSTR. When the computer is in this mode, pressing the STEP lever causes the computer to execute the instruction displayed in the Instruction lights and Stop at t_1 time in the I Cycle of the next instruction. Every time the STEP lever is pressed, the computer executes one more instruction and stops. Every time the computer stops, the I light (Cycle 0) will be on, and the Instruction and Instruction Location lights will describe the next instruction to be executed.

CYCLE BY CYCLE. When the computer is in this mode, pressing the STEP lever causes the computer to execute one cycle of the instruction displayed in the Instruction lights, and Stop at t_1 time of the next cycle. Every time the STEP lever is pressed, the computer executes one more instruction cycle and stops. When the computer stops, the Cycle and Instruction lights will describe the stop point. If the instruction being executed has cycles which repeat (e.g. the shift instructions which repeat Cycle 2 until all the shifts are completed), the computer will stop at each t_1 time, each time the cycle repeats.

The computer will not, however, STEP Cycle by Cycle through an MTP or OPR instruction. This is to prevent possible loss of information because of unusual timing conditions. In these cases, the computer will execute the entire instruction and Stop at t_1 time in the I Cycle of the next instruction. It will not, however, leave the Cycle by Cycle mode.

STEP. The STEP lever is used in the INSTR BY INSTR and CYCLE BY CYCLE modes as described above. When the computer is in neither of these modes, pressing STEP has exactly the same effect as pushing the INSTR BY INSTR button. That is, other console functions will be cleared and the computer will Stop at t_1 time of the next I Cycle. The INSTR BY INSTR light will be on.

ISTOP and XOESTOP. These two pushbuttons are used to stop the computer whenever a reference is made to a selected memory location. The user selects the memory location by setting the Left Switches. The computer will then stop when the Left Switches match the contents of the Memory Address register.

Either button may be pushed when the computer is Running, Paused, or Stopped, although neither will be recognized when the computer is in the CLEAR mode. Neither button changes the state of the computer until the specified stop condition is met. They clear each other and the AUTO RESTART light, but no other console functions. They are themselves cleared only by pressing the STOP lever or the CLEAR pushbutton.

ISTOP. ISTOP will stop the computer at t_1 time when there is a match between the Left Switches and the Memory Address register in the I Cycle. Since in the I Cycle the Memory Address register always holds the location of the instruction, ISTOP is used to stop the computer when it comes to a selected location in the program. In this case the Instruction Location lights and the Memory Address lights will both match the Left Switches when the computer stops. ISTOP is useful for running full speed to a

program trouble spot, at which point the user may wish to STEP Instruction by Instruction or Cycle by Cycle, or to EXAMine a register. If RESUME is pressed the computer will continue normal Running until it again finds a match between the Left Switches and the Memory Address register in the I Cycle.

XOESTOP. The XOESTOP pushbutton will stop the computer at t_1 time when there is an address match in the X, O, or E cycles. When the computer stops, the Memory Address lights will match the Left Switches, and the Cycle lights will display the cycle in which the match was found. The XOESTOP is helpful in tracing the use made of a particular memory register. If, for example, an index register is being incorrectly indexed, or an instruction in the program is being improperly treated as an operand, XOESTOP together with the Left Switches makes it possible to check every reference the program makes to the register in question. XOESTOP will not stop the computer when a match is found during an MTP or OPR instruction.

The STOP Lever

The lever marked STOP should generally be used when other stops (such as STEP, INSTR BY INSTR) are not effective. This lever not only stops the computer, but also does certain presetting of computer control elements. Pushing STOP when the computer is Running will clear all other console functions and interrupt the computer at the next I Cycle, t_1 time. The computer will STOP with the cycle lights cleared. The same is true if STOP is pressed when the computer is Stopped in the X, O, or E Cycles; pressing STOP will cause the computer to Run, finish the current instruction, and then STOP at the next I Cycle, t_1 time with the Cycle lights cleared. Moreover, STOP will interrupt the computer when the computer is Paused, i.e. in an MTP or OPR instruction, without finishing the instruction. Since most console actions cannot be carried out when the computer is Paused, STOP is used to take the computer out of the Pause state and leave it in

the STOP state, a state in which other console actions are effective. Since STOP interrupts a Paused computer without finishing the instruction, it can also be used as a "panic" stop when the user does not want the computer to continue with an MTP or OPR instruction. If, for example, the user observes that the computer is about to write on the wrong block on the tape, or read a tape block into the wrong quarter of memory, STOP can be used in the hope that it will take effect before the transfer takes place.

Pressing STOP clears not only the RUN and CYCLE lights, but also the Memory Contents lights, as well as any console function including ISTOP and XOESTOP. It also turns off the memory flip-flop (MEMFF), the tape write gate (WGFF), and stops any tape motion.

The DO TOG INSTR Lever

DO TOG INSTR. The DO lever is used with the Left and Right Switches to execute instructions manually at the console. Executing instructions at the console should not be confused with the sequential execution of stored programmed instructions which the computer does, for example, when one of the selective starts is used. The latter requires that instructions be stored in the computer memory; however, when instructions are executed at the console with the DO TOG INSTR lever, the computer reads the instruction from the Left and Right toggle Switches. Pressing the DO TOG INSTR lever clears all other console functions except ISTOP and XOESTOP, and interrupts any action the computer might have been carrying out. The computer executes one instruction and STOPS. The instruction must be set in the Left Switches. The Right Switches are used to hold the second word of double register instructions; they are ignored during the execution of single word instructions. When the DO TOG INSTR lever is pressed, the contents of the Left Switches replace the contents of the Instruction and Memory Contents registers (see Instruction Timing Diagrams, Cycle 0).

The Instruction is then executed exactly as though it had come from the memory. The Instruction Location register, however, is never indexed under these circumstances; therefore instructions such as STA i with $\beta = 0$, or SKP, have essentially no effect when executed in this way. The Instruction Location register is changed by Do Tog Instr only when the instruction in the Left Switches is a JMP X; the value X (the right 10 bits of the Left Switches) will replace the contents of the Instruction Location register.

When the computer stops after DO TOG INSTR is used, the Memory Address lights will be cleared, and the Memory Contents lights will display the last operand read from or into the memory during the instruction. If the memory is not used, these lights will be cleared. The Instruction lights will display the instruction just executed, i.e. they will match the Left Switches. The Accumulator will have been changed only if used by the instruction, and the Instruction Location lights will be unchanged (unless the instruction is JMP). Normally, the Cycle lights will display the last cycle of the instruction. If the instruction is MTP, however, the Cycle lights will be left set to the I Cycle. Generally a setting of the X, 0, or E Cycles as indicated in the Cycle lights shows that the current instruction has not been finished. After DO TOG INSTR is used, however, the Cycle lights display the last cycle executed. The DO TOG INSTR lever may be pressed when the computer is Running or Stopped.

Slow Speed Operations

AUTO RESTART and DELAY. By means of these controls, the LINC can be made to run at variable speeds while it is executing stored programmed instructions, or it can be made to repeat certain console functions automatically. The AUTO RESTART pushbutton is always used following some other console action to restart the computer automatically whenever it stops. This generally has the effect of repeating the console function most recently executed.

AUTO RESTART will be recognized any time it is pushed, except when the computer is stopped in the I Cycle of a HLT instruction. It in no way interrupts a Running or Paused computer, nor does it clear any console functions which may be active. It is itself cleared, however, by any other console action, or by executing a HLT instruction. Therefore, to use AUTO RESTART, the computer must first be in the desired mode of operation before pushing the AUTO RESTART button.

The black rotary switch marked DELAY determines the length of time before restart: the lower part of the switch has 4 discrete positions; the upper part is continuous. Together they provide a total range from a few microseconds to 0.9 sec. Changing the setting on the DELAY switch when AUTO RESTART is not lit has no effect.

When AUTO RESTART is pushed after EXAM, or after FILL (which reverts to EXAM), the computer reexamines the register specified by the Left Switches every time the AUTO RESTART DELAY period is over. The user can change the setting of the Left Switches and examine different registers without pushing EXAM again. When used after STEP-EXAM, or after FILL-STEP, (which reverts to STEP-EXAM), the computer will continue stepping the Memory Address register and examining sequentially the contents of the memory. With CLEAR, the computer will restart automatically every time a memory failure causes it to Stop.

When the computer is in CYCLE BY CYCLE or INSTR BY INSTR, AUTO RESTART effectively replaces the STEP lever (without clearing AUTO RESTART), restarting the computer after every stop at t_1 time when the DELAY period is over. With ISTOP or XOESTOP, AUTO RESTART effectively replaces the RESUME lever (without clearing AUTO RESTART). These applications of AUTO RESTART are often useful in monitoring a program.

After a DO TOG INSTR action, AUTO RESTART will usually cause the computer to repeat the last cycle of the instruction, sometimes with interesting effects. After STOP, AUTO RESTART will clear the Memory Address register and examine register 0 in the Memory Contents register.

The Audio Control

The black rotary switch labeled AUDIO is used for aural monitoring of a Running or Paused computer. Like the Delay switch, its lower part has 4 discrete positions and its upper part is continuous. The gong which rings at a HLT instruction or a CLEAR failure is active only when the lower switch is in position 1. The Audio control in no way interacts with the rest of the control console.

Interaction of Pushbuttons and Levers

Some console actions can be made to work simultaneously with other console actions, that is, without clearing one another. These are, namely, STEP, CYCLE BY CYCLE, or INSTR BY INSTR which can be held active during START 20, START 400, START RS, or DO TOG INSTR. This, of course, makes it possible to start a program in the Cycle by Cycle or Instruction by Instruction mode, or to observe a Do Toggle Instruction action cycle by cycle.

To start a program (START 20, START 400, or START RS) in one of the STEP modes, first press the STEP lever. Next, before releasing the STEP lever, push the desired selective start button. The INSTR BY INSTR light will come on and the Instruction Location lights will change to the new starting address. Release the STEP lever and the start pushbutton. The computer will indicate that it is Stopped in the I Cycle of the instruction

at the starting location, in the INSTR BY INSTR mode. The user may now STEP, one instruction at a time, or he may now push CYCLE BY CYCLE, and then STEP one cycle at a time.

The same procedure may be used with the DO TOG INSTR lever. With this lever, however, the user may push either STEP, CYCLE BY CYCLE, or INSTR BY INSTR, before pressing DO TOG INSTR. Again, the first button (or lever) is not released until the DO TOG INSTR lever has been pressed. After both have been released the computer will indicate that it is Stopped in the I Cycle of the toggle instruction. Either the INSTR BY INSTR or CYCLE BY CYCLE light will be on. The user may now STEP through the instruction.

If the toggle instruction is executed CYCLE BY CYCLE, the computer will stop at t_1 time at the beginning of each cycle (except MTP or OPR). At the end of the instruction (after the last cycle has been executed) the computer will bring the next instruction, indicated by the Instruction Location register, into the Instruction and Memory Contents registers, and Stop at t_1 time of the I Cycle of that instruction. It will not stop at the end of the toggle instruction as it does when the DO TOG INSTR lever alone is pressed; it STEPS from the toggle instruction into the next memory instruction. This is also true when the toggle instruction is executed in the INSTR BY INSTR mode.

23 July 1963

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Center Development Office
For Computer Technology in the Biomedical Sciences
292 Main Street, Cambridge 42, Massachusetts

To: LINC Users

From: M. A. Wilkes

Date: 3 September 1963

Re: Tape Generation

It seems that a faulty copy of the Tape Generation program was released to some users. The bad copy generates 257_{10} data words per block instead of 256_{10}. Therefore, please check your binary copies of the MARK program against the enclosed paper copy, and verify that location 165 contains 7426. If it does not (the bad copies have 7425 at location 165), your tapes may well be bad. You may either generate all new tapes or you may check your tapes by the following procedure:

1. CLEAR the memory.
2. FILL locations 0, 377, 400, and 401 with 7777.
3. WRITE AND CHECK Quarter 0 onto some free tape block.
4. CLEAR the memory again.
5. READ AND CHECK into Quarter 0 the tape block just written.
6. EXAM locations 0, 377, and 400. If 0 and 377 contain all ones, and 400 contains all zeros, your tape is all right. If all three locations contain ones, your tape is bad.

If you find you have bad tapes, fix the MARK program, and then generate and check a new tape. Move LAP3 and any other programs you have onto the new tape. (You can use the COPY Meta Command in LAP3 to do this, although other LAP3 routines may give trouble if the LAP3 tape itself is bad.) Then remark all bad tapes.

You will notice that the enclosed paper copy of the MARK program includes a tape WRI and RDC routine to be used after a tape has been marked. You may wish to add these instructions to your copies of the MARK program. Instructions for using both programs are also enclosed.

Tape Mark Program *

Memory Location	Contents		Comment
40	SET i 3	63	
	-1023	6754	Total number of Blocks including extras
	SET i 16	76	
	4	4	
44	SET i 4	64	
	10	10	Initial BN, i.e. -10
	SET i 7	67	
	6	6	
50	SET i 6	66	
	-10	7767	
	MSC 13	13	Write Gate On and MARKUP pulse
	SET i 5	65	
54	0	0	
	SAE i	1460	
	0	0	
	XSK i 5	225	Front End Zone
60	JMP 55	6055	
	SAE i	1460	
	0	0	
	XSK i 6	226	
64	JMP 53	6053	
	SAE i 17	1477	
	SAE i 17	1477	
	SAE i 17	1477	I Marks
70	SAE i 17	1477	
	SAE i 17	1477	
	SAE 16	1456	Forward Block Mark
	SAE i 2	1462	Guard Mark
74	LDA i	1020	

* The SAE instruction operates as a "MARK TAPE instruction" during MARKing. All other instructions operate as described in the LINC ORDER CODE.

Memory Location	Contents		Comments
75	-0	7777	Offset
	SAE i 11	1471	
	ADD 4	2004	
100	SAE i 11	1471	
	COM	17	
	SAE i 11	1471	
	STC 6	4006	
104	SAE i 11	1471	
	SRO 7	1507	
	ADD 5A	2217	
	SAE i 11	1471	
110	SRO 7	1507	
	ADD 5B	2220	
	SAE i 11	1471	
	SRO 7	1507	
114	ADD 5C	2221	
	SAE i 11	1471	
	SRO 7	1507	
	ADD 5D	2222	
120	SAE i 11	1471	
	SRO 7	1507	
	ADD 5E	2223	
	SAE i 11	1471	
124	SRO 7	1507	
	ADD 5F	2224	
	SAE i 11	1471	
	SRO 7	1507	
130	ADD 5G	2225	
	SAE i 11	1471	
	SRO 7	1507	
	ADD 5H	2226	
134	SAE i 11	1471	

Memory Location	Contents		Comments
135	SRO 7	1507	
	ADD 5J	2227	
	SAE i 11	1471	
140	SRO 7	1507	
	ADD 5K	2230	
	SAE i 11	1471	
	SRO 7	1507	
144	ADD 5L	2231	
	SAE i 11	1471	
	SRO 7	1507	
	ADD 5M	2232	
150	SAE i 11	1471	
	SAE i 11	1471	
	SAE i 11	1471	
	STC 6	4006	
154	SAE i 11	1471	
	ADD 4	2004	
	SAE i 11	1471	
	ADA i	1120	
160	-1	7776	
	SAE i 11	1471	
	STC 4	4004	
	SAE i 11	1471	
164	SET i 5	65	
	-351	7426	
	SAE i 11	1471	
	XSK i 5	225	
170	JMP 166	6166	
	SAE i 13	1473	Final Data Mark
	SAE i 1	1461	Check Sum Mark
	SAE i 2	1462	Guard Mark
174	SAE 7	1447	Backward Block Mark

Memory Location	Contents		Comments
175	SAE i 17	1477	I Mark
	XSK i 3	223	
	JMP 66	6066	
200	SAE i	1460	
	0	0	
	SET i 6	66	
	-10	7767	
204	SET i 5	65	
	0	0	
	SAE i	1460	
	0	0	Back End Zone
210	XSK i 5	225	
	JMP 206	6206	
	SAE i	1460	
	0	0	
214	XSK i 6	226	
	JMP 204	6204	
	HLT	0	Clear MARKFF
	#5A 10	10	
220	#5B 4	4	
	#5C 2	2	
	#5D 1	1	
	#5E 200	200	
224	#5F 100	100	
	#5G 40	40	
	#5H 20	20	
	#5J 4000	4000	
230	#5K 2000	2000	
	#5L 1000	1000	
232	#5M 400	400	

Tape Write and Check Routine

Memory Location	Contents		Comments
300	LDA i	1020	
	3/0	3000	
	STA	1040	
	313	313	
304	STC 324	4324	
	SET i 1	61	Clear Quarter 3
	1377	1377	
	STA i 1	1061	
310	XSK 1	201	
	JMP 307	6307	
	WRI i u	736	WRI zeros in every Block
	[3/0]	~	
314	LDA i	1020	
	1	1	
	ADM	1140	
	313	313	
320	SAE i	1460	
	4000	4000	
	JMP 312	6312	
	RDC i u	730	RDC every Block
324	[3/0]	~	
	ADD 315	2315	
	ADM	1140	
	324	324	
330	SAE i	1460	
	4000	4000	
	JMP 323	6323	
	MTB i u	733	Move toward Block 0 to rewind the tape, and HLT
334	0	0	
335	HLT	0	